SUPERVISORY CONTROL

Pedro U. Lima

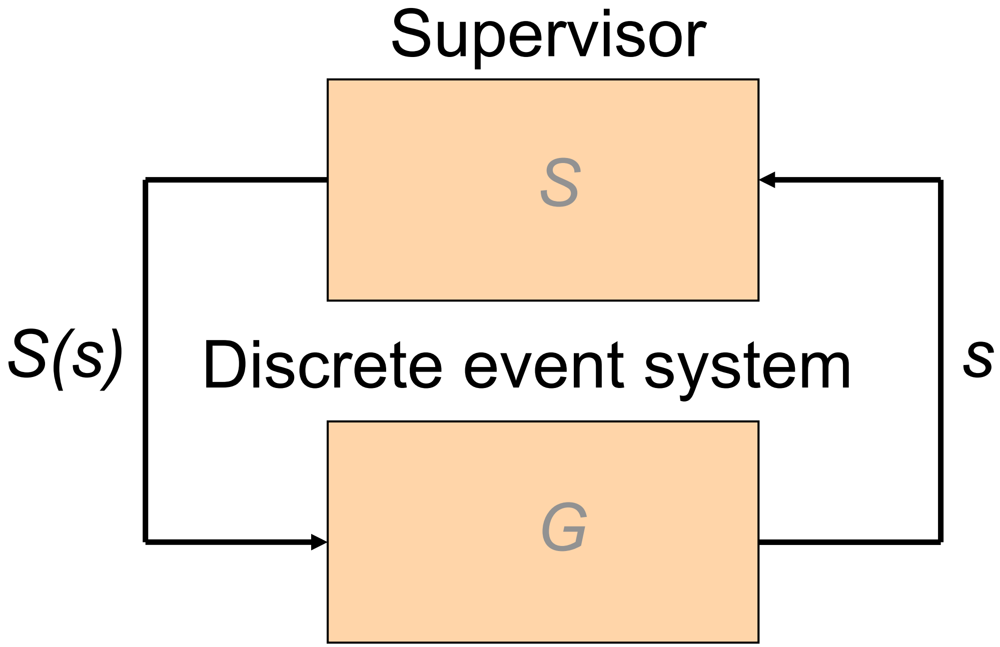
Instituto Superior Técnico (IST)
Instituto de Sistemas e Robótica (ISR)
Av.Rovisco Pais, 1
1049-001 Lisboa
PORTUGAL

October 2002
Revised November 2003

Supervision of DES

Basic Notions: Dynamic Feedback Supervision
and Admissible Behaviors
Controllability
Dealing with Blocking
Modular Control
Observability
Decentralized Control

SUPERVISORY CONTROL – AN INTRODUCTION



- **What do we mean by specifications ?**
- **How does *S* modify the behavior of *G* ?**

FEEDBACK CONTROL WITH SUPERVISORS

CONTROLLED D.E.S.

DES G : $G = (X, E, f, \Gamma, x_0, X_m)$, X may be infinite

Language of DES G : $L(G) = L$, $L = \overline{L}$

Marked language of G: $L_m(G) = L_m$

Controllable events : E_c

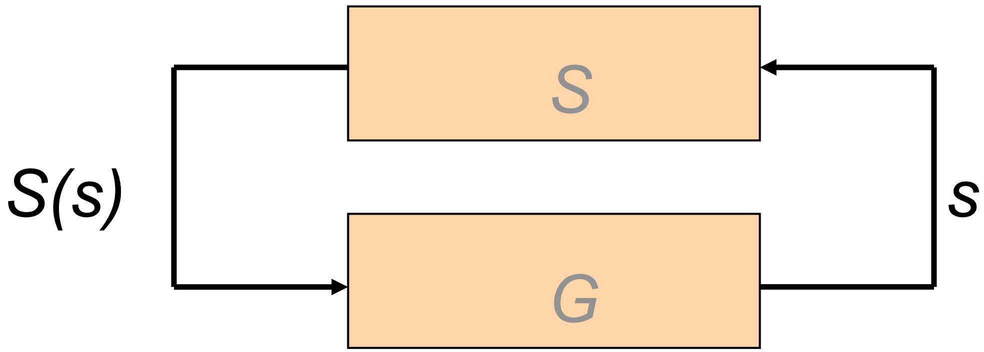
Uncontrollable events: E_{uc}

(e.g., faults, high priority events, hardware or actuation limitations)

$$L_m \subseteq L, \qquad E = E_c \cup E_{uc}$$

FEEDBACK CONTROL WITH SUPERVISORS

SUPERVISOR D.E.S.



Control policy *S*

Control action *S(s)*

Supervisor function : $S : L(G) \rightarrow 2^E$

Enabled transitions : $S(s) \cap \Gamma(f(x_0, s))$

FEEDBACK CONTROL WITH SUPERVISORS

SUPERVISOR D.E.S. (cont'd)

Admissible supervisors :

$$\forall_{s \in L(G)} \; E_{uc} \cap \Gamma\big(f(x_0, s)\big) \subseteq S(s)$$

S is not allowed to ever disable a feasible uncontrollable event.

The feedback loop *S/G* ("*S* controlling *G*") is an instance of *dynamic* feedback since the domain of *S*(.) is *L(G)* and not *X.* Thus the control action may change in subsequent visits to the same state $x \in X$.

LANGUAGES GENERATED AND MARKED BY S/G

LANGUAGE GENERATED BY S/G

1. $\varepsilon \in L(S/G)$
2. $\left[\left(s \in L(S/G)\right) \wedge \left(s\sigma \in L(G)\right) \wedge \left(\sigma \in S(s)\right)\right] \Leftrightarrow \left[s\sigma \in L(S/G)\right]$

LANGUAGE MARKED BY S/G

$$L_m(S/G) = L(S/G) \cap L_m(G)$$

$$\emptyset \subseteq L_m(S/G) \subseteq \overline{L_m(S/G)} \subseteq L(S/G) \subseteq L(G)$$

LANGUAGES GENERATED AND MARKED BY S/G

$$L(S/G) = \overline{L(S/G)}$$ - prefix closed by definition

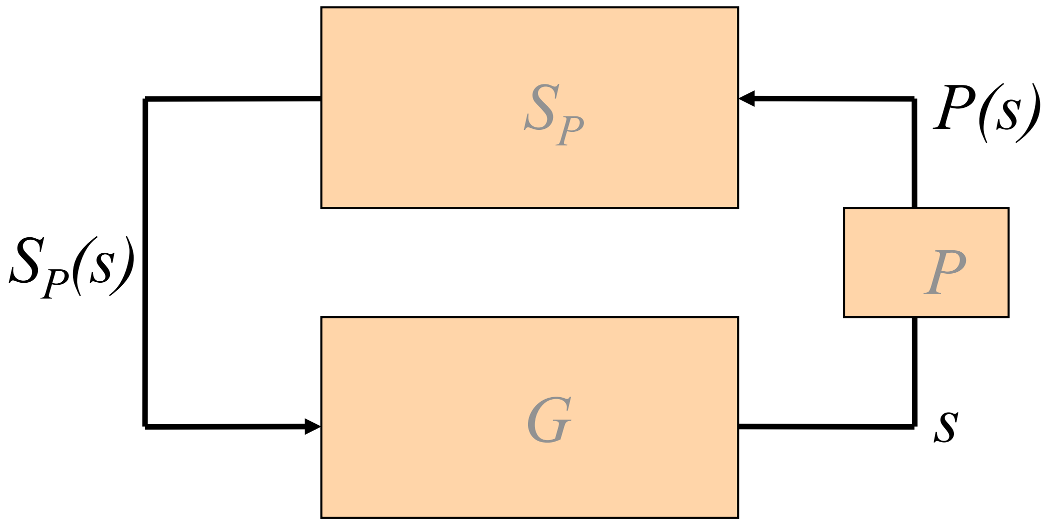
DES S/G is blocking : $L(S/G) \neq \overline{L_m(S/G)}$

DES S/G non blocking : $L(S/G) = \overline{L_m(S/G)}$

DES S/G blocking $\Rightarrow$ supervisor *S* is blocking

DES S/G non blocking $\Rightarrow$ supervisor *S* is non blocking

CONTROL UNDER PARTIAL OBSERVATION



Observable and unobservable events E_o, E_{uo}

$$E = E_o \cup E_{uo}$$

$$S_P : P[L(G)] \to 2^E$$

CONTROL UNDER PARTIAL OBSERVATION

The projection $P : E^* \rightarrow E_0^*$ hides the unobservable events executed by G from P - supervisor S_P

- The supervisor cannot distinguish between two strings s_1 and s_2 with the same projection, i.e., $P(s_1) = P(s_2)$.
- For such $s_1, s_2 \in L(G)$, the P-supervisor will issue the same control action, $S_P[P(s_1)]$.
- The control action can change only after the occurrence of an observable event, i.e., when $P(s)$ changes.

Assumption: when an (enabled) observable event occurs, the control action is *instantaneously* updated, ***before*** any unobservable event occurs.

CONTROL UNDER PARTIAL OBSERVATION

Assume $t = t'\sigma$ is observed and define

$$L_t = P^{-1}(t')\{\sigma\}(S_P(t) \cap E_{uo})^* \cap L(G), \; \sigma \in E_o$$

L_t contains all the strings in *L(G)* that are effectively subject to the control action $S_P(t)$, when S_P *controls* G, i.e., those belonging to $P^{-1}(t')\{\sigma\}$ as well as to the unobservable continuation of $P^{-1}(t')\{\sigma\}$

Admissible
P-supervisors :

$$\forall_{t = t'\sigma \in P[L(G)]} \; E_{uc} \cap \Big[\bigcup_{s \in L_t} \Gamma(f(x_0, s))\Big] \subseteq S_P(t)$$

S_P is not allowed to ever disable a feasible (but possibly unobservable) uncontrollable continuation in *L(G)* of all strings that S_P applies to. **Note that the control action remains in effect until the next observable event is executed by *G*.**

LANGUAGES GENERATED AND MARKED BY S_P/G

LANGUAGE GENERATED BY S_P/G

1. $\varepsilon \in L(S_P / G)$

2. $[(s \in L(S_P / G)) \wedge (s\sigma \in L(G)) \wedge (\sigma \in S_P[P(s)])] \Leftrightarrow [s\sigma \in L(S_P / G)]$

LANGUAGE MARKED BY S_P/G

$$L_m(S_P / G) = L(S_P / G) \cap L_m(G)$$

Note that $L(S_P/G)$ and $L_m(S_P/G)$ are defined over E, and not E_o, corresponding to the closed-loop behavior of G before the effect of the projection of P.

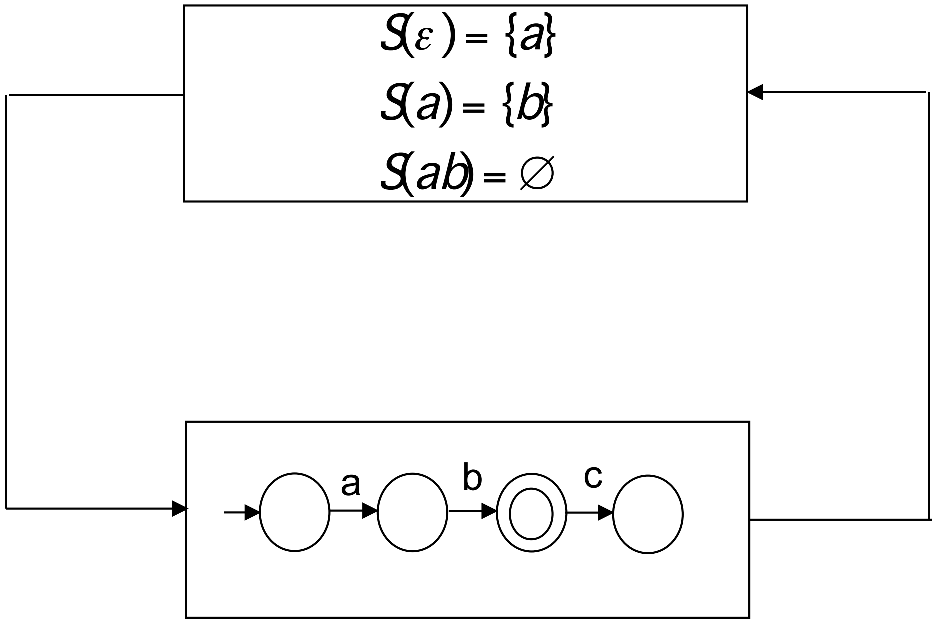
FEEDBACK CONTROL WITH SUPERVISOR
An Example

$L(G) = \overline{\{abc\}}$

$L_m(G) = \{ab\}$

G is blocking

all events controllable and observable



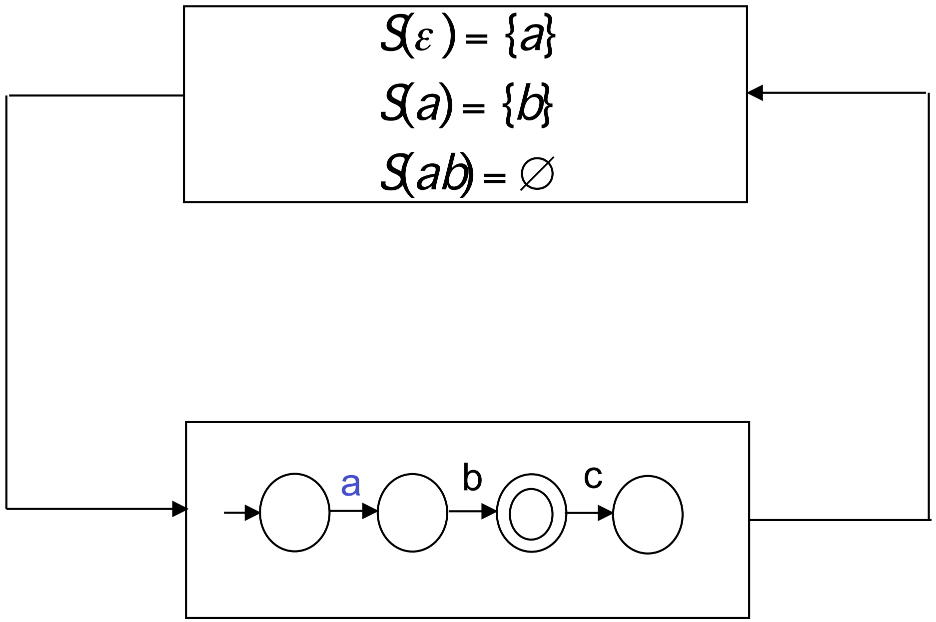
FEEDBACK CONTROL WITH SUPERVISOR
An Example

$L(G) = \overline{\{abc\}}$

$L_m(G) = \{ab\}$

G is blocking

all events controllable and observable



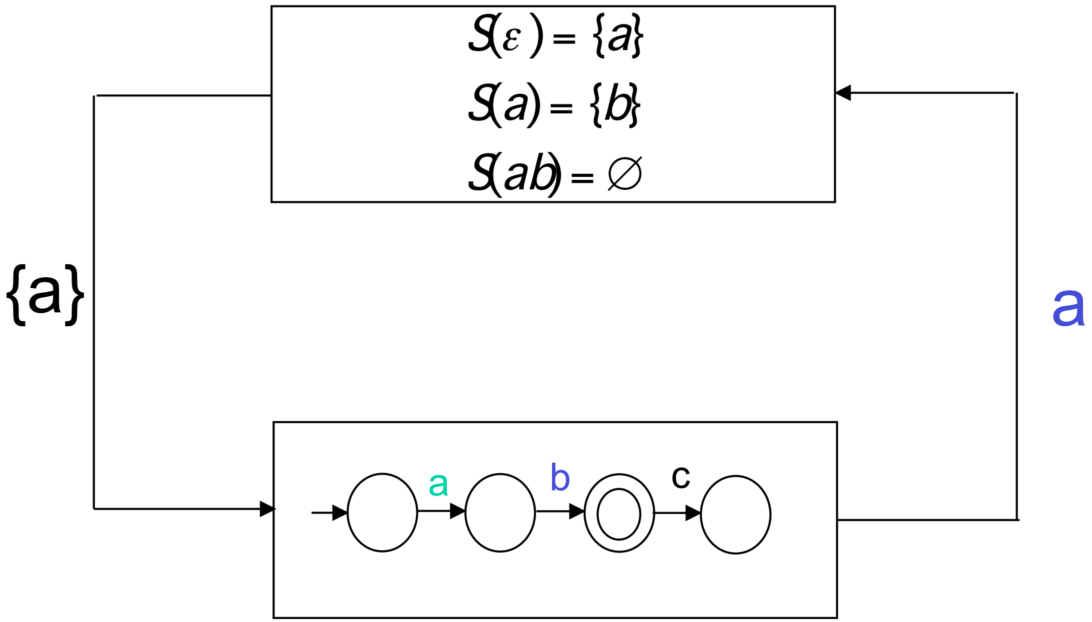
FEEDBACK CONTROL WITH SUPERVISOR
An Example

$$L(G) = \overline{\{abc\}}$$

$$L_m(G) = \{ab\}$$

G is blocking

all events controllable and observable



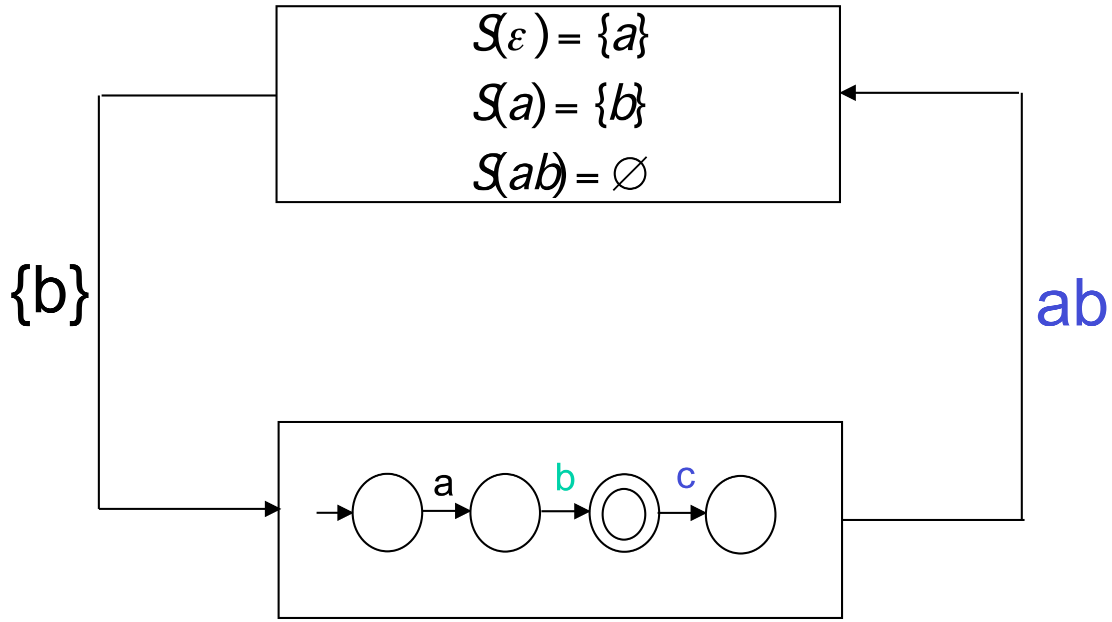
FEEDBACK CONTROL WITH SUPERVISOR
An Example

$L(G) = \overline{\{abc\}}$

$L_m(G) = \{ab\}$

G is blocking

all events controllable and observable



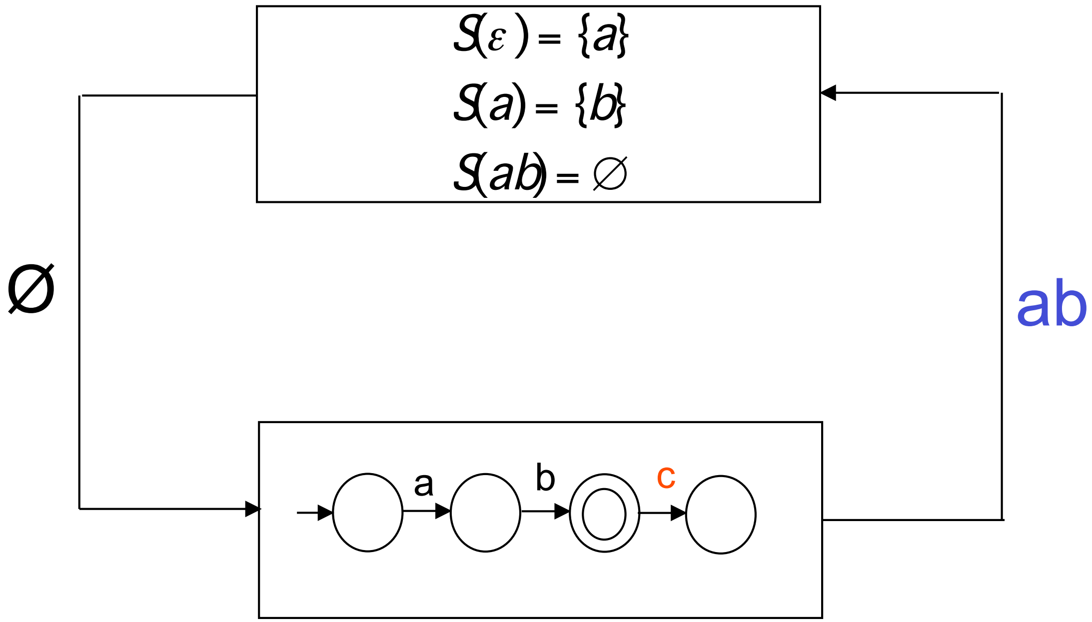
FEEDBACK CONTROL WITH SUPERVISOR
An Example

$L(G) = \overline{\{abc\}}$

$L_m(G) = \{ab\}$

G is blocking

all events controllable and observable



$L(S/G) = \overline{L_m(S/G)} = \overline{\{ab\}}$

S/G is nonblocking

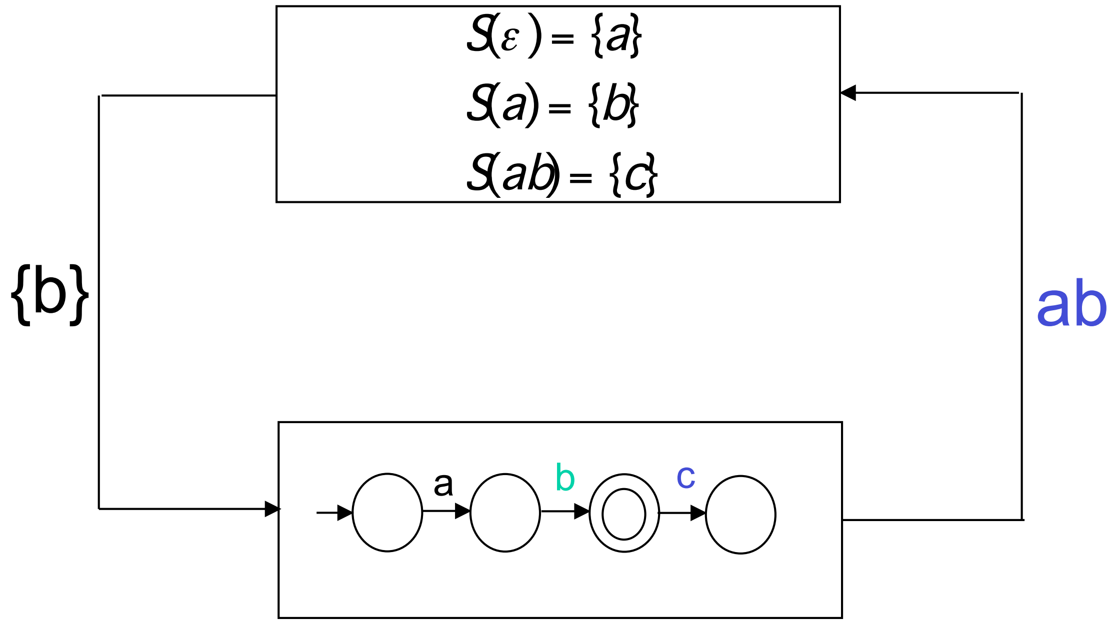
FEEDBACK CONTROL WITH SUPERVISOR
An Example

$$L(G) = \overline{\{abc\}}$$

$$L_m(G) = \{ab\}$$

G is blocking

event c uncontrollable



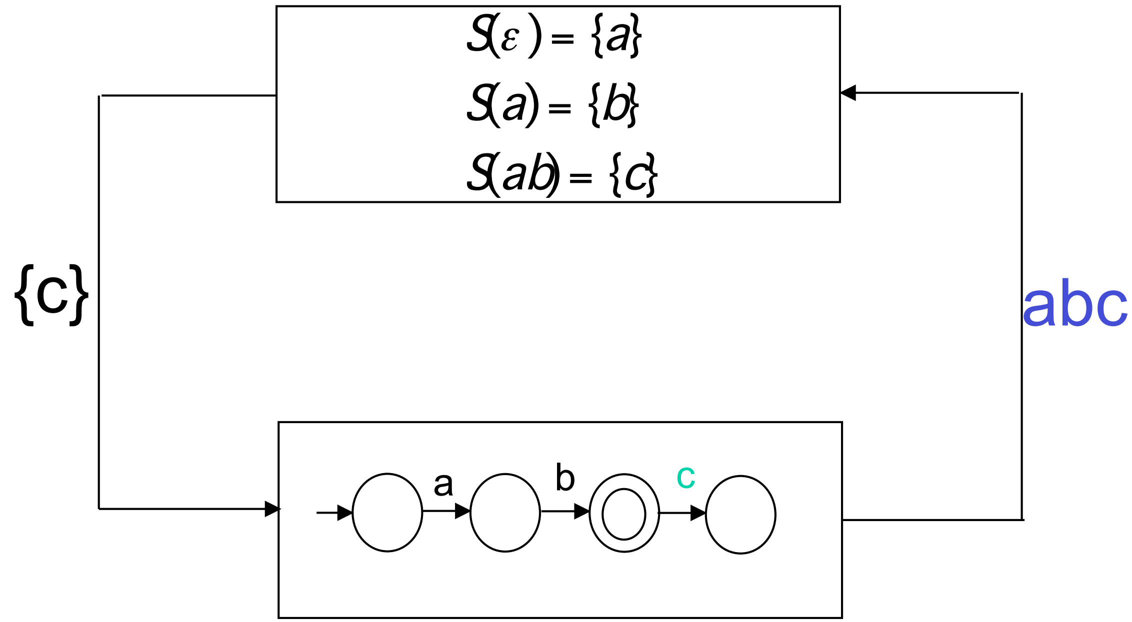
FEEDBACK CONTROL WITH SUPERVISOR
An Example

$L(G) = \overline{\{abc\}}$

$L_m(G) = \{ab\}$

G is blocking

event c uncontrollable



$L(S/G) \neq \overline{L_m(S/G)}$

S/G is blocking

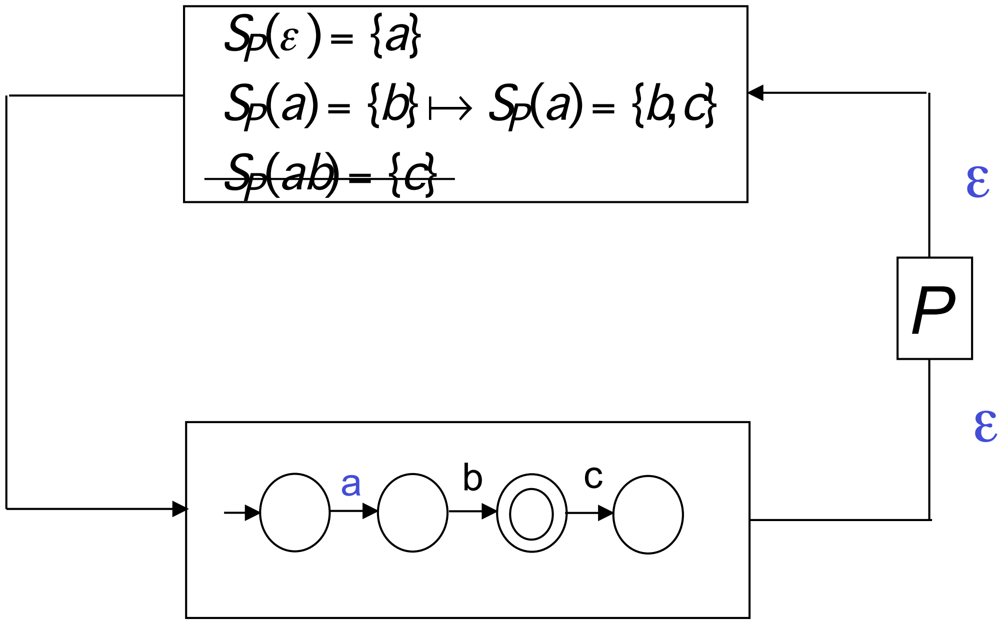
FEEDBACK CONTROL WITH SUPERVISOR
An Example

$L(G) = \overline{\{abc\}}$

$L_m(G) = \{ab\}$

G is blocking

$E_{uo}=\{b\}$ $E_{uc}=\{c\}$



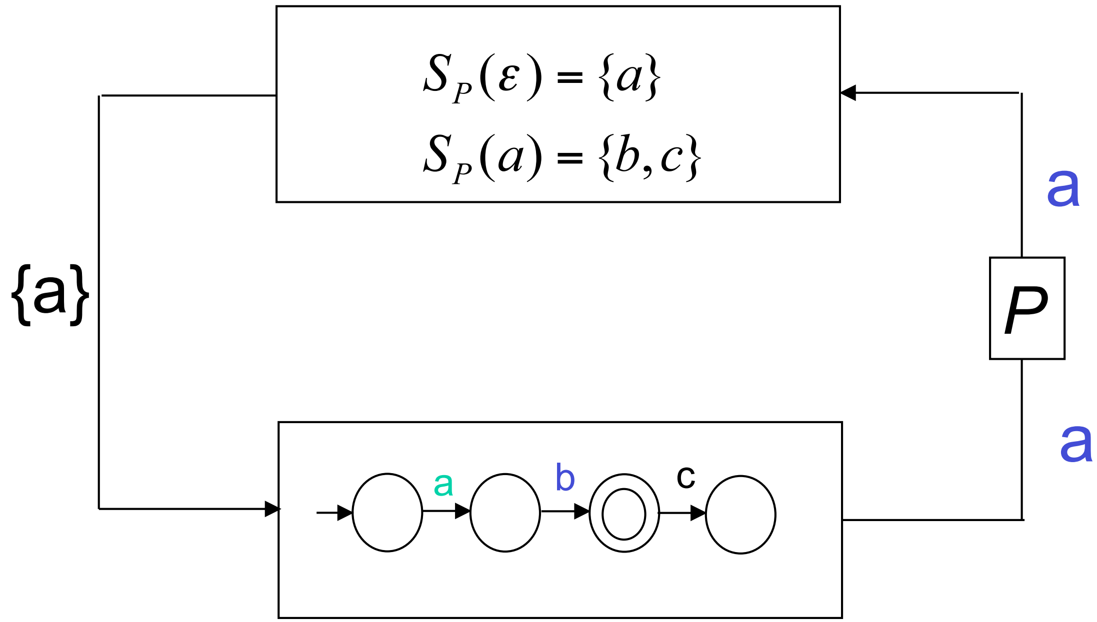
FEEDBACK CONTROL WITH SUPERVISOR
An Example

$L(G) = \overline{\{abc\}}$

$L_m(G) = \{ab\}$

G is blocking

E_{uo}={b} E_{uc}={c}



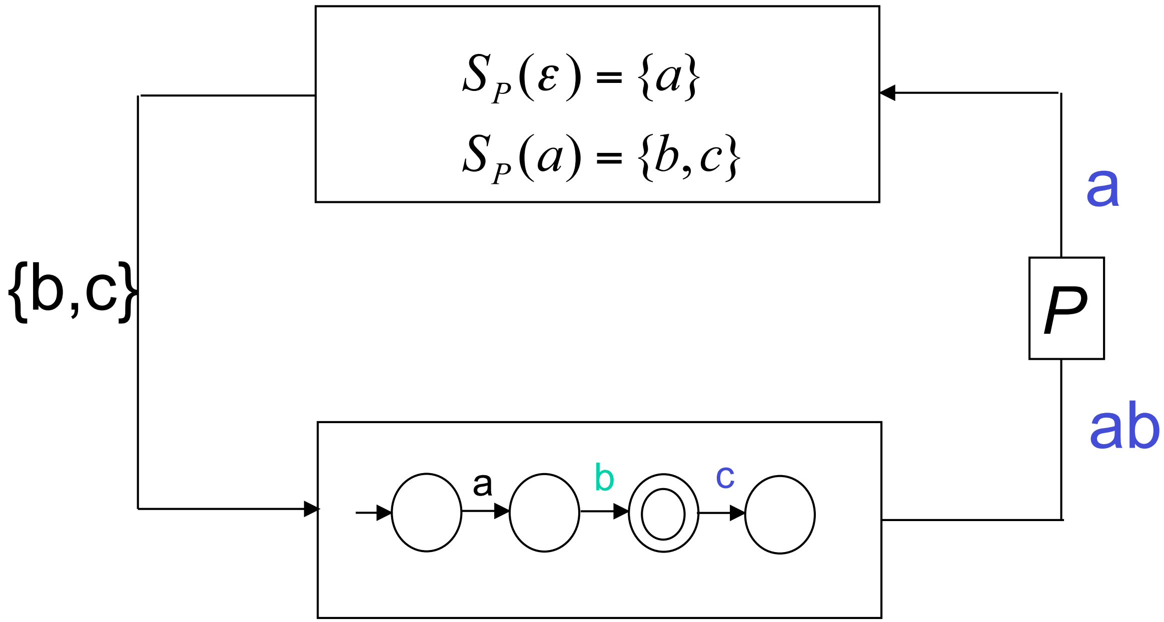
FEEDBACK CONTROL WITH SUPERVISOR
An Example

$L(G) = \overline{\{abc\}}$

$L_m(G) = \{ab\}$

G is blocking

$E_{uo}=\{b\}$ $E_{uc}=\{c\}$



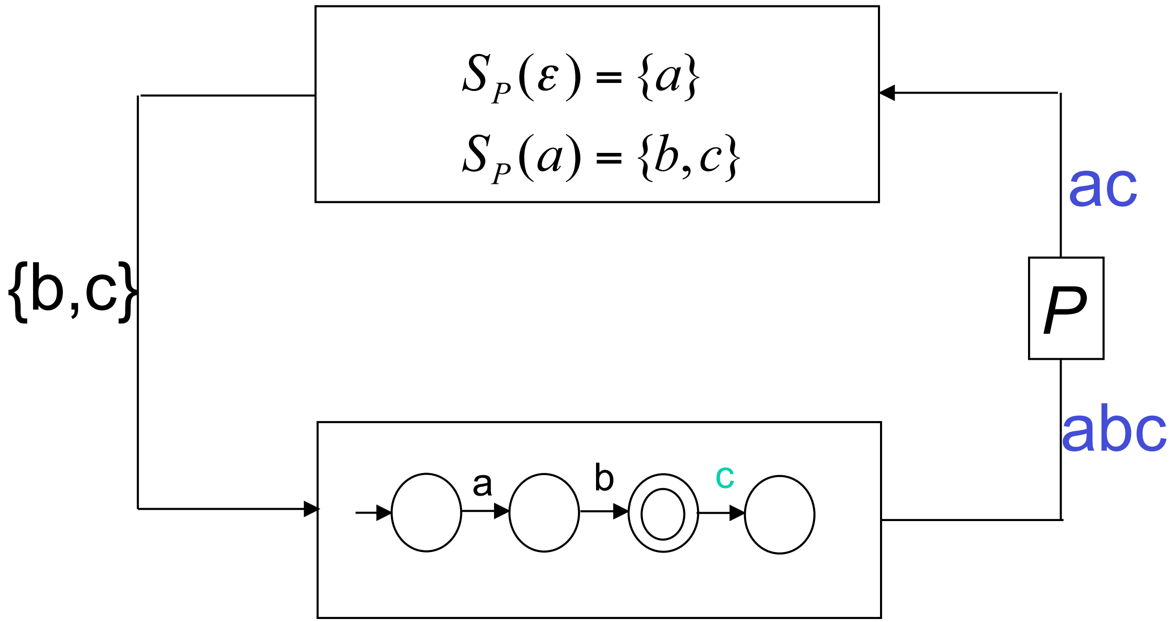
FEEDBACK CONTROL WITH SUPERVISOR
An Example

$L(G) = \overline{\{abc\}}$

$L_m(G) = \{ab\}$

G is blocking

E_{uo}=\{b\} ***E_{uc}=\{c\}***



$L(S/G) \neq \overline{L_m(S/G)}$

S/G is blocking

SPECIFICATIONS

Required (marked) language : $L_r \; (L_{rm})$

(minimal required behavior)

Admissible (marked) language : $L_a \, (L_{am})$

(maximal admissible behavior)

$$L_r \subseteq L(S/G) \subseteq L_a \subset L(G)$$

$$L_{rm} \subseteq L_m(S/G) \subseteq L_{am} \subset L_m(G)$$

For partial-observation problems, S is replaced by S_P.
When blocking is a concern, we focus on ensuring $L_m(S/G) \subseteq L_{am}$ as well as mitigating blocking.

$$\text{Assumption}: \; L_a = \overline{L_a}$$

In the sequel, we will consider all languages regular.

AUTOMATON MODEL OF SPECIFICATIONS

Combination of H_{spec} and G *to obtain* H_a such that $L(H_a) = L_a$

This is valid for other language requirements as well.

In this case, we say that H_a is a *recognizer* of L_a.

- If the events that appear in G but not in H_{spec} are **irrelevant** to the specifications that H_{spec} implements, then we use *parallel composition*
- If the events are absent from H_{spec} because they should **not** happen in the admissible language L_a, then we use *product composition*

AUTOMATON MODEL OF SPECIFICATIONS
Example: Illegal States

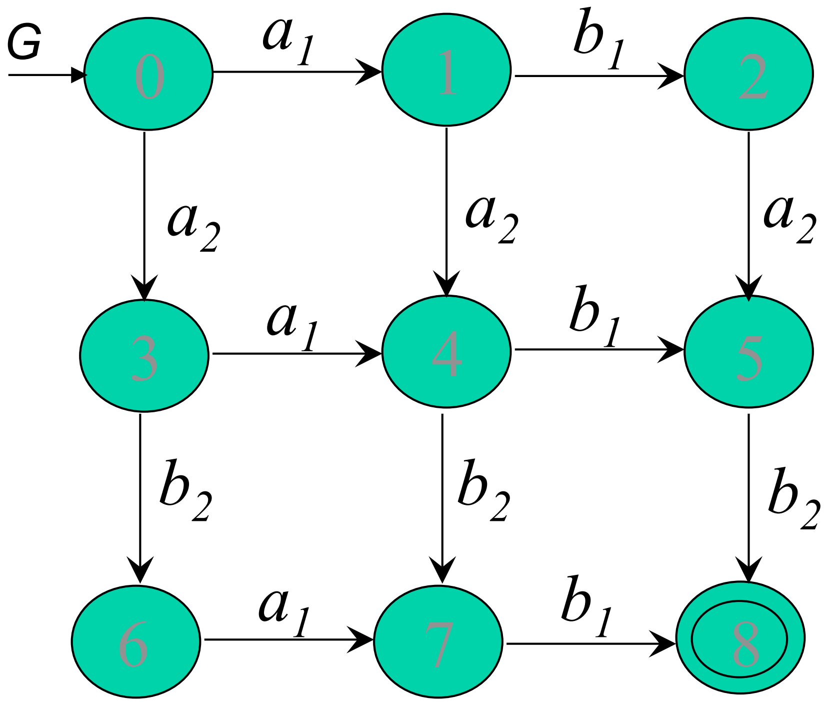
1. delete illegal states from *G*, by removing the states and the transitions attached to them, obtaining *G'*;
2. $H_a = Ac\,(G')$
3. $L(H_a) = L_a$

If the specification also requires nonblocking behavior

- delete illegal states from *G*, by removing the states and the transitions attached to them, obtaining *G'*;
- $H_a = Trim\,(G')$
- $L_m(H_a) = L_{am}$ and $L(H_a) = \overline{L}_{am}$

AUTOMATON MODEL OF SPECIFICATIONS
Example: State Splitting

If a specification requires remembering how a particular state of *G* was reached in order to determine what future behavior is admissible, then that state must be split into as many states as necessary. The active event set of each newly introduced state is adjusted according to the respective admissible continuations.

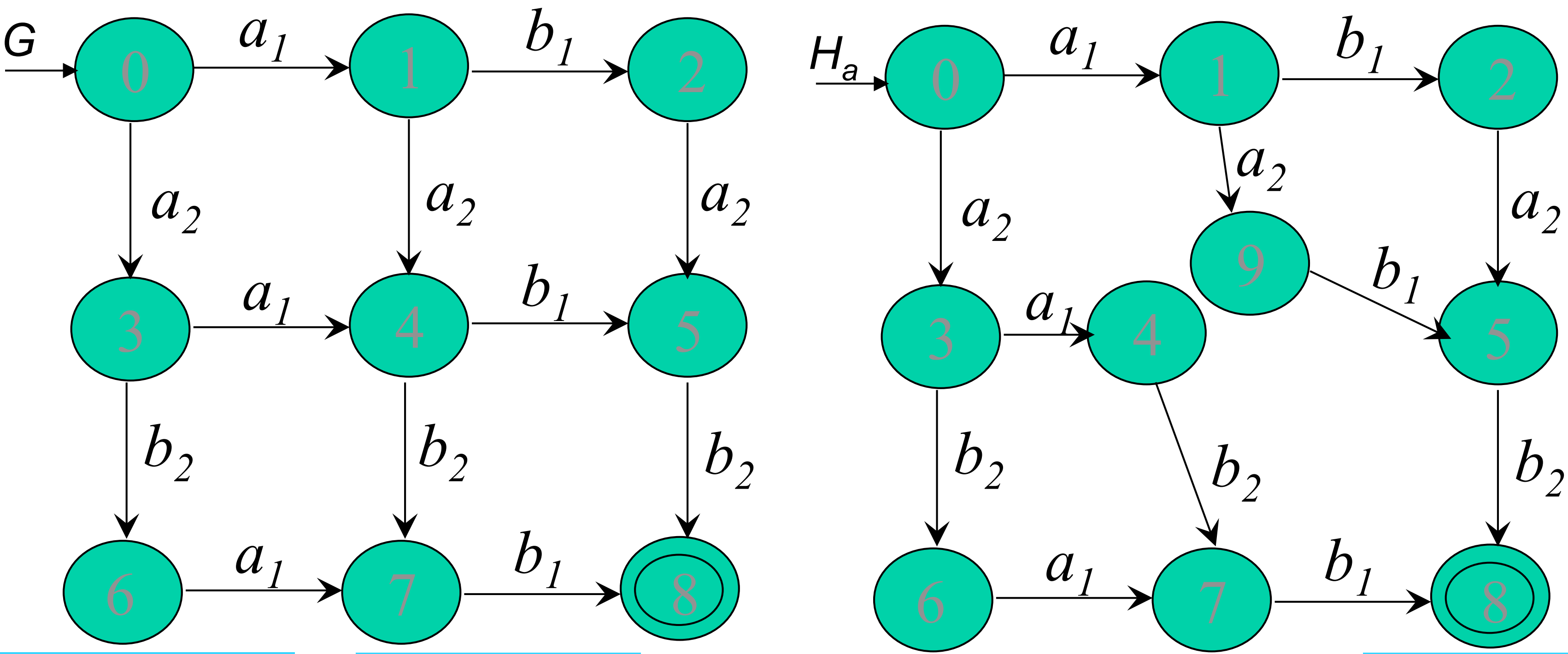


Example: database concurrency control problem with *T1=a1b1* and *T2=a2b2* .

- *L(G)* contains inadmissible strings (or schedules).
- The only admissible strings are those where *a1* precedes *a2* iff *b1* precedes *b2*.

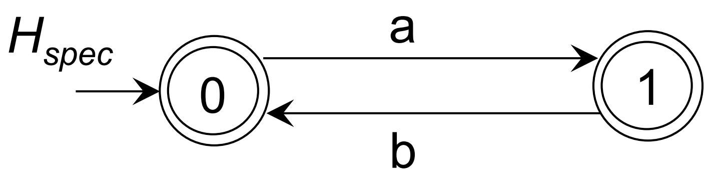
AUTOMATON MODEL OF SPECIFICATIONS
Example: State Splitting

The *trim* automaton H_a is such that *$L_m(H_a)$ contains only the admissible strings of $L_m(G)$* and is also nonblocking.



AUTOMATON MODEL OF SPECIFICATIONS
Example: Event Alternance

If a specification requires the alternance of two events (e.g., *a* and *b*, with *a* being the first event to occur)

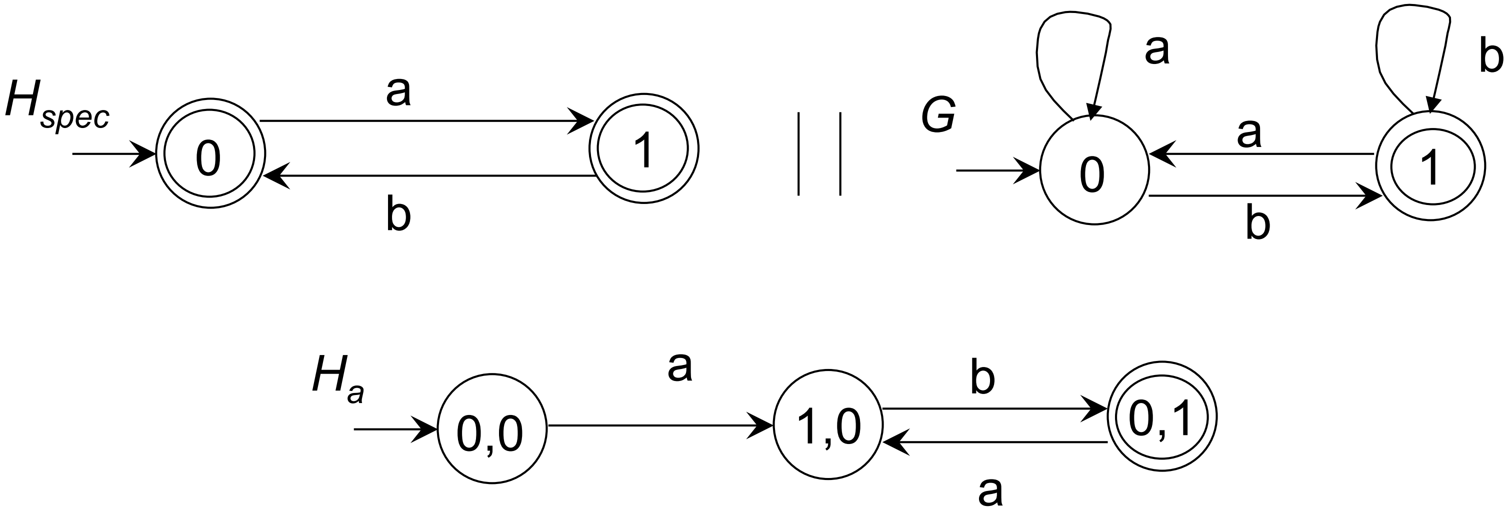


$$H_a = H_{spec} \| G$$

Both states of H_{spec} are marked since the specification does not involve blocking; therefore, marking in H_a is consistent with marking in G.

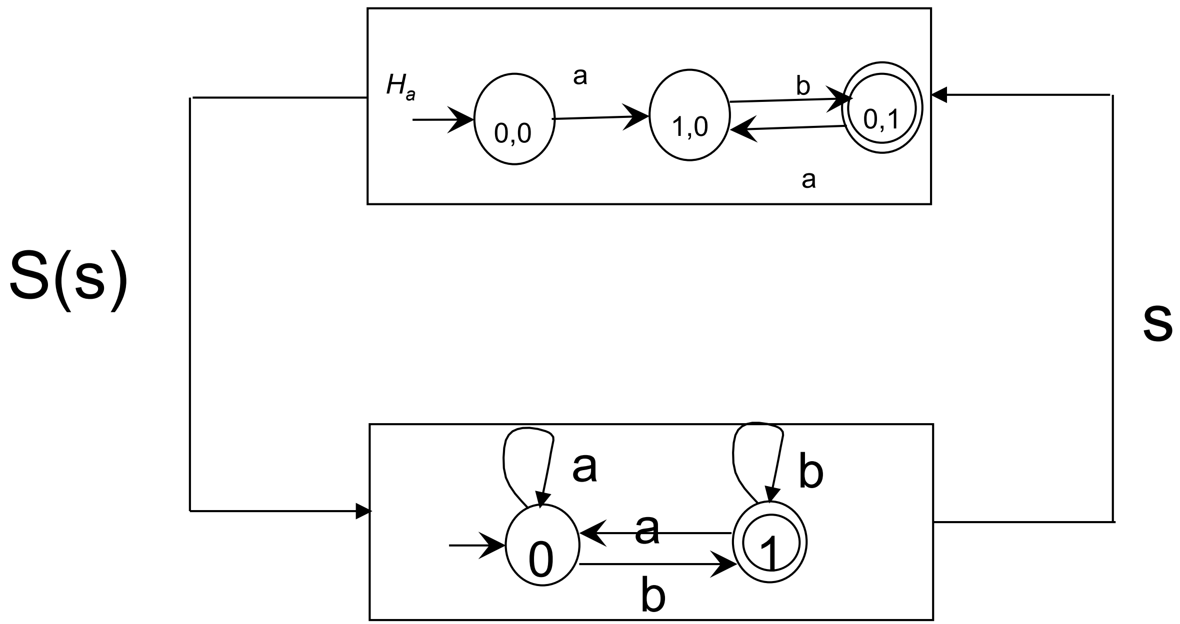
AUTOMATON MODEL OF SPECIFICATIONS
Example: Event Alternance

$$H_a = H_{spec} \,\|\, G$$



Both states of H_{spec} are marked since the specification does not involve blocking; therefore, marking in H_a is consistent with marking in G.

AUTOMATON MODEL OF SPECIFICATIONS
Example: Event Alternance



Ex.: aaaabbaabab in L(G) ➔ ababab in L(S/G)

AUTOMATON MODEL OF SPECIFICATIONS
Example: Illegal Substring

If a specification identifies as illegal all strings of $L(G)$ that contain substring $s_f = \sigma_1 \ldots \sigma_n \in E^*$

we build $H_{spec} = (X, E, f, x_0, X)$ *as* follows:

1. $X = \{\varepsilon, \sigma_1, \sigma_1\sigma_2, \ldots, \sigma_1\ldots\sigma_{n-1}\}$
(i.e., we associate a state of H_{spec} to every proper prefix of s_f)

2. (a) $f(\sigma_1\ldots\sigma_i, \sigma_{i+1}) = \sigma_1\ldots\sigma_i\sigma_{i+1}$, for $i=0,\ldots,n-2$.
(b) Complete f to E (except for state $\sigma_1\ldots\sigma_{n-1}$, completed to $E \setminus \{\sigma_n\}$, since σ_n is *illegal* in that state:
$f(\sigma_1\ldots\sigma_i, \gamma)$= state in X corresponding to the longest suffix of $\sigma_1\ldots\sigma_i\gamma$

3. Take $x_0 = \varepsilon$

$$L(H_{spec}) = L_m(H_{spec}) = E^* \setminus \{\text{strings having } s_f \text{ as substring}\}$$

$$H_a = H_{spec} \times G$$

CONTROLLABILITY THEOREM

Given a DES G with $E_{uc} \subseteq E$ and a specification language $K \subseteq L(G),\ K \neq \emptyset$

There exists supervisor S such that $L(S/G) = \overline{K}$

iff

$$\overline{K}E_{uc} \cap L(G) \subseteq \overline{K}$$ (controllability condition)

> *" If you cannot prevent it, then it should be legal "*

Proof is constructive: $S(s) = [E_{uc} \cap \Gamma(f(x_0, s))] \cup \{\sigma \in E_c : s\sigma \in \overline{K}\}$

DEFINITION OF CONTROLLABILITY

Given $E_{uc} \subseteq E$,

$M = \overline{M}$ and K languages over event set E

If $\overline{K}E_{uc} \cap M \subseteq \overline{K}$

Then *K is controllable with respect to M and E_{uc}*

Controllability is a property of the prefix-closure of a language, thus *K* is controllable iff $\overline{K}$ is controllable.

Language expression: $\forall_{s \in \overline{K}} \forall_{e \in E_{uc}}, se \in M \Rightarrow se \in \overline{K}$

REALIZATION OF SUPERVISORS

If $K \subseteq L(G)$ is controllable, the Controllability Theorem tells us that the supervisor *S* defined by

$$S(s) = [E_{uc} \cap \Gamma(f(x_0, s))] \cup \{\sigma \in E_c : s\sigma \in \overline{K}\}$$

results in $L(S/G) = \overline{K}$, excluding $\overline{K} = L(G)$ and $\overline{K} = \emptyset$.

How do we build a convenient representation of *S*?

• domain of *S* can be restricted to $L(S/G) = \overline{K}$.
• *G* is an automaton – we use also an FSA to represent *S* (this is called a *realization* of *S*)

We will be dealing with regular languages *L(G)* and *K*, with finite, thus implementable, realizations.

REALIZATION OF SUPERVISORS

To build an automaton realization of *S*, we just build an automaton *R* that marks the language $\overline{K}$.

$$R=(Y,E,g,\Gamma_R,y_0,Y)$$

$$L_m(R)=L(R)=\overline{K}.$$

Note that

$$L(R\times G)=L(R)\cap L(G)=\overline{K}\cap L(G)=\overline{K}=L(S/G)$$

$$L_m(R\times G)=L_m(R)\cap L_m(G)=\overline{K}\cap L_m(G)=L(S/G)\cap L_m(G)=L_m(S/G)$$

and also that $R \| G=R\times G$, since *R* and *G* share the same event set *E*. This means that *S(s)* is encoded in the transition structure of *R*:

$$S(s)=[E_{uc}\cap\Gamma(f(x_0,s))]\cup\left\{\sigma\in E_c : s\sigma\in\overline{K}\right\}$$ → from the controllability of *K*

$$=\Gamma_R(g(y_0,s))=\Gamma_{R\times G}(g\times f((y_0,x_0),s))$$ → from $\overline{K}\subseteq L(G)$

REALIZATION OF SUPERVISORS

How is *S* implemented?

1. Let *G* be in state *x* and *R* be in state *y*, following the execution of string $s \in L(S/G)$.
2. *G* generates an event σ that is currently enabled. This means that this event is also present in the active event set of *R* at *y*.
3. Thus *R* also executes the event, as a passive observer of *G*.
4. Let *x'* and *y'* be the new states of *G* and *R* after the execution of σ. The set of enabled events of *G* after string $s\sigma$ is now given by the active event set of *R* at *y'*.

REALIZATION OF SUPERVISORS

Induced Supervisors

Q: If we are given automaton *C* and form the product *C*×*G*, can that be interpreted as controlling *G* by some supervisor?

$$S_i^C(s) = \begin{cases} \left[E_{uc} \cap \Gamma(f(x_0, s))\right] \cup \left[\sigma \in E_c : s\sigma \in L(C)\right] & \text{if } s \in L(G) \cap L(C) \\ E_{uc} & \text{otherwise} \end{cases}$$

A: $L(S_i^C/G) = L(C \times G)$ **iff** $L(C)$ is controllable w.r.t. $L(G)$ and E_{uc}.

THE PROPERTY OF CONTROLLABILITY

Suppose uncontrollable language $\overline{K}$:

$$\overline{K}E_{uc} \cap M \not\subseteq \overline{K} \text{ w.r.t. } M = \overline{M} \subseteq E^* \text{ and } E_{uc} \subseteq E$$

We assume $K \subseteq M$, but we do not assume K to be prefix - closed

$K^{\uparrow C}$ is the supremal controllable sublanguage of K

$K^{\downarrow C}$ is the infimal prefix - closed and controllable superlanguage of K

$$\emptyset \subseteq K^{\uparrow C} \subseteq K \subseteq \overline{K} \subseteq K^{\downarrow C} \subseteq M$$

Properties of controllability

1. If K_1 and K_2 are controllable, then $K_1 \cup K_2$ is controllable.
2. If K_1 and K_2 are controllable, then $K_1 \cap K_2$ need not be controllable.
3. If $\overline{K_1} \cap \overline{K_2} = \overline{(K_1 \cap K_2)}$ and K_1 and K_2 are controllable, then $K_1 \cap K_2$ is controllable.
4. If K_1 and K_2 are prefix-closed and controllable, then $K_1 \cap K_2$ is prefix-closed and controllable.

THE PROPERTY OF CONTROLLABILITY

Nonconflicting languages

Languages K_1 and K_2 are said to be *nonconflicting* if they satisfy the condition

$$\overline{K_1} \cap \overline{K_2} = \overline{(K_1 \cap K_2)}$$

Intuitive meaning: *if K_1 and K_2 share a prefix, then they must share a string containing that prefix.*

- Note that $\overline{K_1} \cap \overline{K_2} \supseteq \overline{(K_1 \cap K_2)}$ always holds.
- Prefix-closed languages satisfy the above condition.

THE PROPERTY OF CONTROLLABILITY

class of controllable sublanguages of *K*

$$C_{in}(K) = \left\{ L \subseteq K : \overline{L}E_{uc} \cap M \subseteq \overline{L} \right\}$$

class of prefix-closed and controllable superlanguages of *K*

$$CC_{out}(K) = \left\{ L \subseteq E^* : (K \subseteq L \subseteq M) \text{ and } (\overline{\mathrm{L}} = \mathrm{L}) \text{ and } (\overline{L}E_{uc} \cap M \subseteq \overline{L}) \right\}$$

$$\emptyset \in C_{in}(K) \text{ and } M \in CC_{out}(K)$$

SUPREMAL CONTROLLABLE SUBLANGUAGE

Existence

We would like to find the “largest” sublanguage of K which is controllable.
Q: Does it exist?
A: Yes!

$$K^{\uparrow C} = \bigcup_{L \in C_{in}(K)} L$$

By definition, $L \subseteq K^{\uparrow C}$, for any $L \in C_{in}(K)$ $\Rightarrow$ $K^{\uparrow C}$ is the *supremal controllable sublanguage* of K.

- In the “worst” case, $K^{\uparrow C} = \varnothing$
- If K is controllable, then $K^{\uparrow C} = K$

SUPREMAL CONTROLLABLE SUBLANGUAGE

Properties

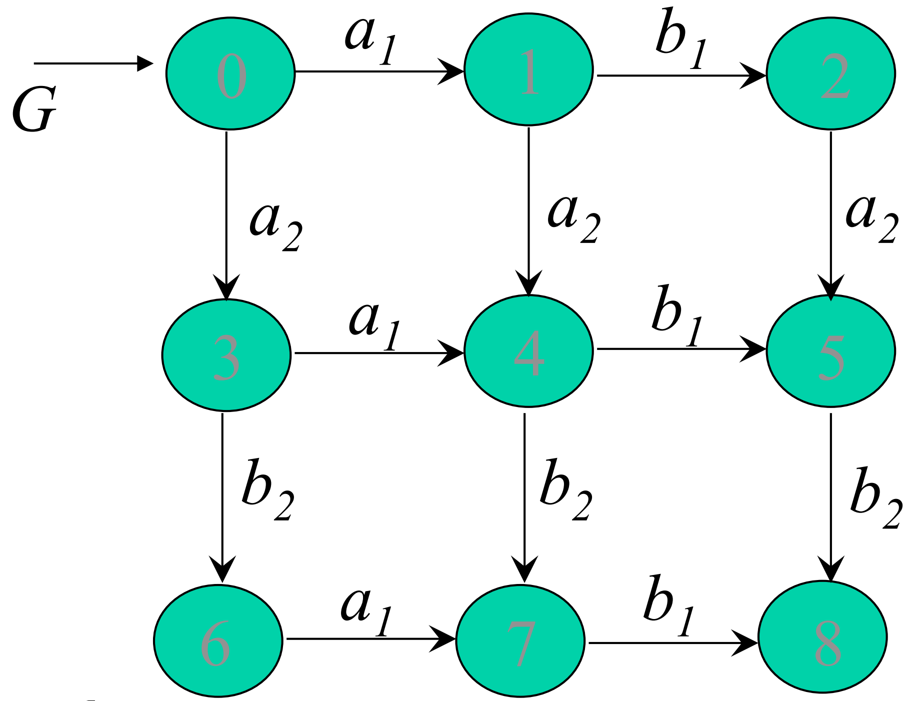
$$K_1 \subseteq K_2 \Rightarrow K_1^{\uparrow C} \subseteq K_2^{\uparrow C}$$

Proposition: If K is prefix-closed, so is $K^{\uparrow C}$.

Proposition (properties of the $\uparrow C$ operation):

1. $(K_1 \cap K_2)^{\uparrow C} \subseteq K_1^{\uparrow C} \cap K_2^{\uparrow C}$
2. $(K_1 \cap K_2)^{\uparrow C} = \left(K_1^{\uparrow C} \cap K_2^{\uparrow C}\right)^{\uparrow C}$
3. If K_1 and K_2 are non-conflicting, then $(K_1 \cap K_2)^{\uparrow C} = K_1^{\uparrow C} \cap K_2^{\uparrow C}$
4. $(K_1 \cup K_2)^{\uparrow C} \supseteq K_1^{\uparrow C} \cup K_2^{\uparrow C}$

Example of supremal controllable sublanguage



$$K = \{a_2 b_2 a_1 b_1, a_2 a_1 b_2 b_1, a_1 a_2 b_1 b_2, a_1 b_1 a_2 b_2\}$$

$$L(G) = M,\ E_{uc} = \{a_2, b_2\},\ K^{\uparrow C} = ?$$

K is not controllable (w.r.t. M and E_{uc}): $\longleftrightarrow$ $a_1 a_2 \in \overline{K}$ can be extended in M by the uncontrollable event b_2, and $a_1 a_2 b_2 \notin \overline{K}$

Example of supremal controllable sublanguage (cont'd)

Removing from K all strings that contain $a_1 a_2$ as a prefix, we get the language

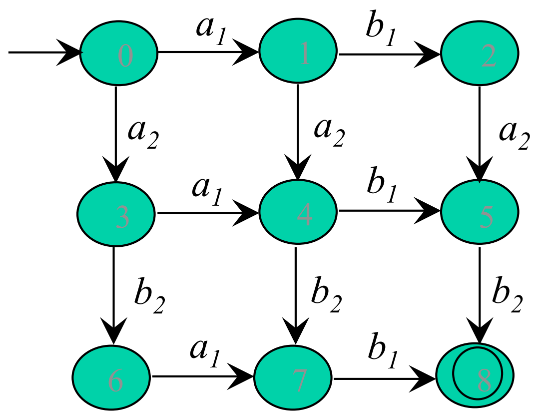
$$K_1 = \{a_2 b_2 a_1 b_1, a_2 a_1 b_2 b_1, a_1 b_1 a_2 b_2\}$$

K_1 is not controllable: $a_1 \in \overline{K}_1$ can be extended in M by the uncontrollable event a_2, and $a_1 a_2 \notin \overline{K}_1$

Removing now from K_1 all strings that contain a_1 as a prefix, we get the language

$$K_2 = \{a_2 b_2 a_1 b_1, a_2 a_1 b_2 b_1\}$$

$$K^{\uparrow C} = K_2$$



INFIMAL PREFIX-CLOSED CONTROLLABLE SUPERLANGUAGE

Existence

We would like to find the “smallest” superlanguage of *K* which is controllable.

Q: Does it exist?

A: Yes!

$$K^{\downarrow C} = \bigcap_{L \in CC_{out}(K)} L$$

By definition, $K^{\downarrow C} \subseteq L$, for any $L \in CC_{out}(K)$ $\Rightarrow$

$K^{\downarrow C}$ is the *infimal prefix-closed controllable superlanguage* of *K*.

- In the “worst” case, $K^{\downarrow C} = M$
- If *K* is controllable, then $K^{\downarrow C} = K$

INFIMAL PREFIX-CLOSED CONTROLLABLE SUPERLANGUAGE

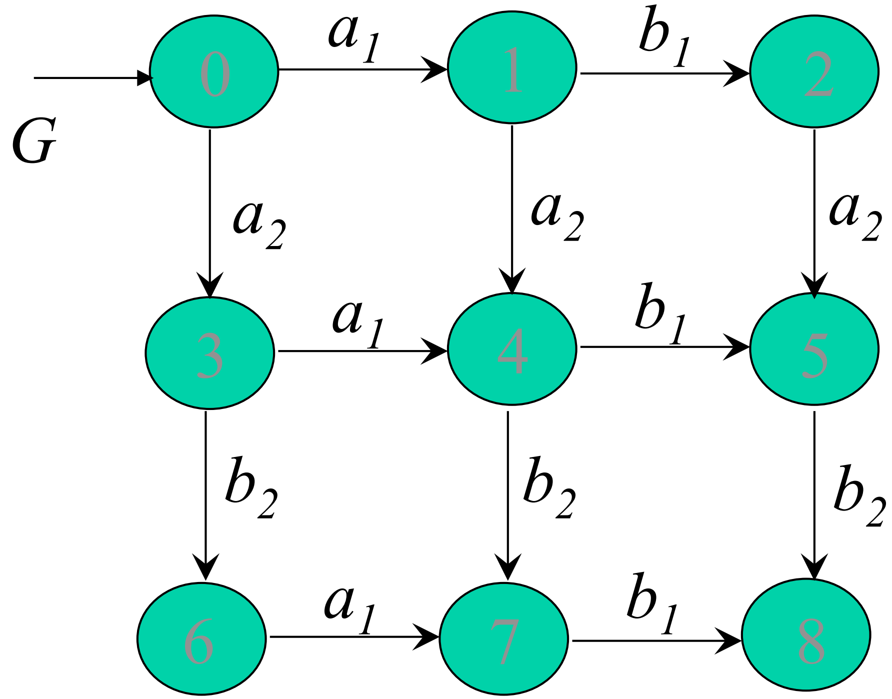
Properties

$$K_1 \subseteq K_2 \Rightarrow K_1^{\downarrow C} \subseteq K_2^{\downarrow C}$$

Proposition (properties of the $\downarrow C$ operation):

1. $(K_1 \cap K_2)^{\downarrow C} \subseteq K_1^{\downarrow C} \cap K_2^{\downarrow C}$
2. If K_1 and K_2 are non-conflicting, then $(K_1 \cap K_2)^{\downarrow C} = K_1^{\downarrow C} \cap K_2^{\downarrow C}$
3. $(K_1 \cup K_2)^{\downarrow C} \supseteq K_1^{\downarrow C} \cup K_2^{\downarrow C}$

Example of infimal prefix-closed controllable superlanguage



$$K = \{a_2b_2a_1b_1, a_2a_1b_2b_1, a_1a_2b_1b_2, a_1b_1a_2b_2\}$$

$$L(G) = M,\ E_{uc} = \{a_2, b_2\}$$

Solution to make $\overline{K}$ controllable is to extend $a_1\, a_2$ with a string of uncontrollable events of length one $K^{\downarrow C} = \overline{K} \cup \{a_1a_2b_2\}$.

SOME SUPERVISORY CONTROL PROBLEMS

Typically we want

$$\varnothing \subseteq L_r \subseteq L_r^{\downarrow C} \subseteq L(S/G) \subseteq L_a^{\uparrow C} \subseteq L_a \subset L(G)$$

This is the *range problem*, for L_r and L_a prefix-closed languages.
The problem has solution only if $L_r \subseteq L_a^{\uparrow C}$.

We will investigate next two particular cases of this.
We are not concerned with *blocking* yet.

SOME SUPERVISORY CONTROL PROBLEMS

for a DES *G* with event set *E* and $E_{uc} \subseteq E$ and $L_a = \overline{L}_a \subseteq L(G)$.

Basic Supervisory Control Problem (BSCP)

Find a supervisor *S* such that:

1. $L(S \mid G) \subseteq L_a$
2. $L(S \mid G)$ is "the largest it can be", i.e., for any other supervisor S_{other} such that $L(S_{other} \mid G) \subseteq L_a$, $L(S_{other} \mid G) \subseteq L(S \mid G)$

Solution: $L(S \mid G) = L_a^{\uparrow C}$

The behavior of *G* is restricted in order to stay inside the admissible behavior, but no more than necessary. L_a is obtained from *L(G)* by removing illegal states in *G* and illegal strings in *L(G)*.

The solution is optimal with *set inclusion* as the criterion of optimality. The optimal solution contains all other solutions (*minimally restrictive*).

SOME SUPERVISORY CONTROL PROBLEMS

for a DES G with event set E and $E_{uc} \subseteq E$ and $L_r \subseteq L(G)$

Dual of Basic Supervisory Control Problem (DuSCP)

Find a supervisor S such that:

1. $L(S \mid G) \supseteq L_r$
2. $L(S \mid G)$ is "the smallest it can be", i.e., for any other supervisor S_{other} such that $L(S_{other} \mid G) \supseteq L_r$, $L(S_{other} \mid G) \supseteq L(S \mid G)$

Solution: $L(S \mid G) = L_r^{\downarrow C}$

In a range problem, the behavior of G is restricted in order to be the smallest solution inside the range. Again, the essence of the control problem is to handle the presence of uncontrollable events.

The solution is optimal with *set inclusion* as the criterion of optimality.
The optimal solution is contained in all other solutions (*maximally restrictive*).

SOME SUPERVISORY CONTROL PROBLEMS

for a DES G with event set E and $E_{uc} \subseteq E$

desired language $L_{des} \subseteq L(G)$ and tolerated language $L_{tol} = \overline{L_{tol}} \subseteq L(G)$

where $\overline{L_{des}} \subseteq L_{tol}$

Supervisory Control Problem with Tolerance (SCPT)

Find a supervisor S such that:

1. $L(S/G) \subseteq L_{tol}$ — *S/G* can never exceed the tolerated language
2. for all prefix-closed and controllable $K \subseteq L_{tol}$,
   $K \cap L_{des} \subseteq L(S/G) \cap L_{des}$ — *S/G* to achieve as much of L_{des} as possible
3. $K \cap L_{des} = L(S/G) \cap L_{des} \Rightarrow L(S/G) \subseteq K$ — achieve 2. with the smallest possible *L(S/G)*

Solution: $L(S/G) = (L_{tol}^{\uparrow C} \cap L_{des})^{\downarrow C}$

by 1. by 3. by 2.

The idea is to achieve as much as possible of the desired language without ever exceeding the tolerated language. Unlike in the range problem, we allow not achieving all of L_{des}, as long as we achieve as much of it as possible. Think of L_{des} as the solution to adopt if all events were controllable.

NONBLOCKING CONTROLLABILITY THEOREM

Specifications on the controlled system are now given as a sublanguage of $L_m(G)$, and S is required to be nonblocking, i.e.,

$$\overline{L_m(S/G)} = L(S/G)$$

Given a DES G with $E_{uc} \subseteq E$ and a specification language $K \subseteq L_m(G)$, $K \neq \varnothing$

There exists a *nonblocking* supervisor S such that $L_m(S/G) = K$ and $L(S/G) = \overline{K}$

iff

$$\begin{cases} \overline{K}E_{uc} \cap L(G) \subseteq \overline{K} & \text{(controllability condition)} \\ K = \overline{K} \cap L_m(G) & (L_m(G)\text{-closure}) \end{cases}$$

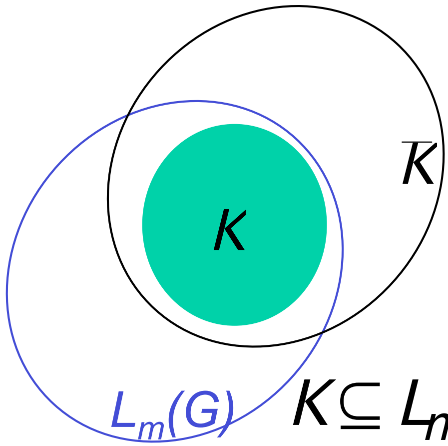
NONBLOCKING CONTROLLABILITY THEOREM

proof is again constructive - same supervisor as for the CT:

$$S(s) = [E_{uc} \cap \Gamma(f(x_0, s))] \cup \{\sigma \in E_c : s\sigma \in \overline{K}\}$$

$L_m(G)$-closure condition:

$K \subseteq \overline{K} \cap L_m(G)$ **always holds.**



$K \subseteq L_m(G),\ K \subseteq \overline{K}$

$K \supseteq \overline{K} \cap L_m(G)$ **may not hold.**

Example :

$$L_m(G) = \{\alpha_1, \alpha_1\beta_1\alpha_1, \alpha_1\beta_1\alpha_1\beta_1\alpha_1\}$$

$$K = \{\alpha_1\beta_1\alpha_1\}$$

K violates the $L_m(G)$-closure condition since it does not contain string α_1.

PROPERTIES OF THE ↑C OPERATION

$L_m(G)$-closure condition typically holds *by construction of K*, when *K* is interpreted as "admissible marked behavior". Some supporting arguments:

- *marking* is a property of the uncontrolled system *G*
- specifications are usually stated in terms of prefix-closed languages $K_{spec} = \overline{K_{spec}}$
- the admissible marked language is $K = K_{spec} \cap L_m(G)$
- such a *K* is guaranteed to be $L_m(G)$-closed.

so we will assume that any "admissible marked behavior" satisfies the $L_m(G)$-closure condition and will be concerned with the controllability condition only.

Proposition (further properties of the ↑*C* operation):

1. If $K \subseteq L_m(G)$ is $L_m(G)$ - closed, then so is $K^{\uparrow C}$
2. In general, $\overline{K^{\uparrow C}} \subseteq (\overline{K})^{\uparrow C}$

NONBLOCKING SUPERVISORY CONTROL

for a DES *G* with event set *E*
and $E_{uc} \subseteq E$ and $L_{am} \subseteq L_m(G)$, with L_{am} assumed to be $L_m(G)$-closed

Basic Supervisory Control Problem - Nonblocking (BSCP-NB)

Find a nonblocking supervisor *S* such that:

1. $L_m(S/G) \subseteq L_{am}$
2. $L_m(S/G)$ is "the largest it can be", i.e., for any other supervisor S_{other} such that $L_m(S_{other}/G) \subseteq L_{am}$, $L_m(S_{other}/G) \subseteq L_m(S/G)$

Solution: $L(S/G) = \overline{L_{am}^{\uparrow C}}$ and $L_m(S/G) = L_{am}^{\uparrow C}$, $L_{am}^{\uparrow C} \neq \emptyset$.

This is the *minimally restrictive nonblocking* solution.

NONBLOCKING SUPERVISORY CONTROL

Note that

$$L_{am} = \overline{L_{am}} \cap L_m(G) \Rightarrow L_{am}^{\uparrow C} = \overline{L_{am}^{\uparrow C}} \cap L_m(G)$$

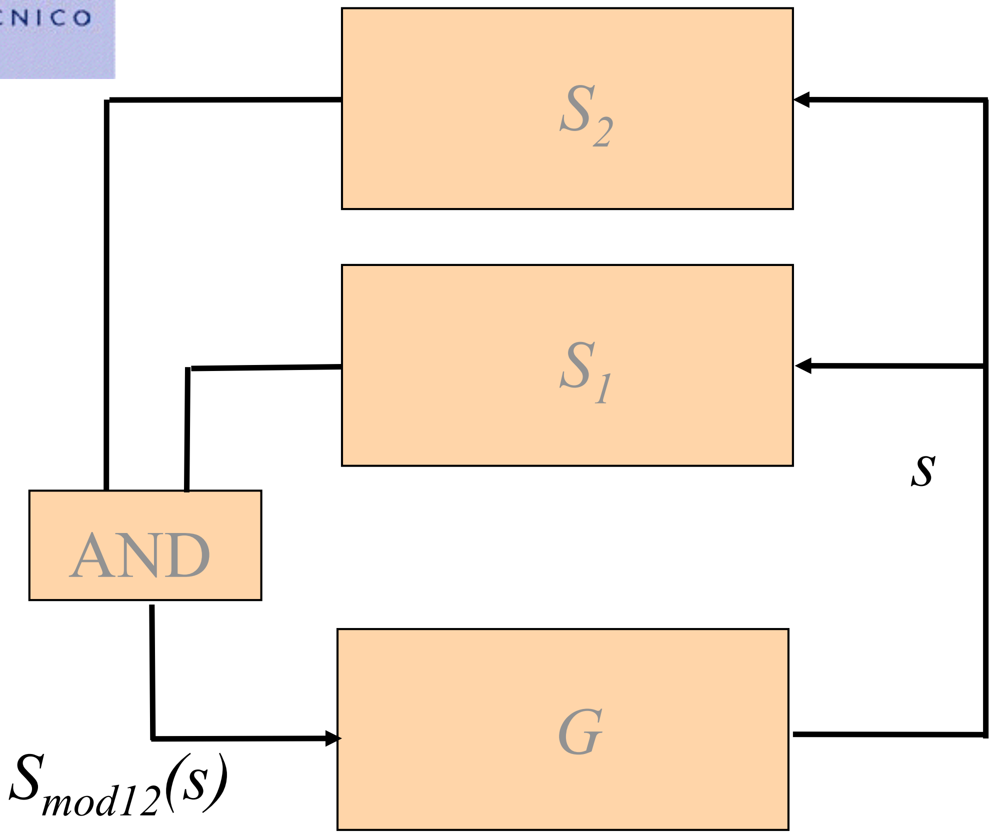
guarantees that

$$L_m(S \mid G) = L_{am}^{\uparrow C} \text{ whenever}$$

$$L(S \mid G) = \overline{L_{am}^{\uparrow C}}.$$

Hence, *S* can be realized by building a recognizer of $\overline{L_{am}^{\uparrow C}}$

MODULAR CONTROL



$$S_{\mathrm{mod}12}(s) = S_1(s) \cap S_2(s)$$

$$L(S_{\mathrm{mod}\ 12}/G) = L(S_1/G) \cap L(S_2/G)$$

$$L_m(S_{\mathrm{mod}\ 12}/G) = L_m(S_1/G) \cap L_m(S_2/G)$$

Note: Given standard realizations R_1 and R_2 of S_1 and S_2, respectively, the standard realization of S_{mod12} could be obtained by building $R=R_1 x R_2$. However, we may need to store as many as $n_1 n_2$ states.

Using S_{mod12} we can still interpret the supervision of G by S_{mod12} as $R_1 x R_2 x G$, but only $n_1 + n_2$ states must be stored.

MODULAR SUPERVISORY CONTROL PROBLEM

for a DES *G* with event set *E* and $E_{uc} \subseteq E$ and

admissible language $L_a = L_{a1} \cap L_{a2}$, where $L_{ai} = \overline{L_{ai}} \subseteq L(G),\ i = 1,2$

Modular Supervisory Control Problem (MSCP)

Find a modular supervisor S_{mod} such that

$$L_{\text{mod}}(S \mid G) = L_a^{\uparrow C}$$

which is the same as what can be achieved by BSCP (*monolithic* approach)

Solution: $L(S_i \mid G) = L_{ai}^{\uparrow C},\ i = 1,2$ and then take

$$S_{\text{mod}}(s) = S_{\text{mod}12}(s) = S_1 \cap S_2$$

$$L(S_{\text{mod}} \mid G) = L_{a1}^{\uparrow C} \cap L_{a2}^{\uparrow C} = (L_{a1} \cap L_{a2})^{\uparrow C} = L_a^{\uparrow C}$$

This holds because the L_{ai}s are prefix-closed

MODULAR SUPERVISORY CONTROL PROBLEM

The same simple approach does not necessarily work in general for the *nonblocking* version of MSCP.

Proposition (nonblocking modular supervisors):
Let S_i, i=1,2, be individual nonblocking supervisors for G. Then S_{mod12} is nonblocking **iff** $L_m(S_1/G)$ and $L_m(S_2/G)$ are nonconflicting languages, that is, if and only if

$$\overline{L_m(S_1/G) \cap L_m(S_2/G)} = \overline{L_m(S_1/G)} \cap \overline{L_m(S_2/G)}.$$

Implication: if we consider $L_{am} = L_{am1} \cap L_{am2}$, where $L_{ami} = \overline{L_{ami}} \subseteq L_m(G)$, and each L_{ami} is $L_m(G)$-closed, $i = 1,2$ ($\Rightarrow L_{am}$ is $L_m(G)$-closed), the intuitive approach of first synthesizing S_i such that $L(S_i \mid G) = \overline{L_{ami}^{\uparrow C}}$ and then forming S_{mod12} yelds:

K is prefix-closed, so is $K^{\uparrow C}$

property 1. of $\uparrow C$

$$L(S_{\mathrm{mod}12}/G) = \overline{L_{am1}^{\uparrow C}} \cap \overline{L_{am2}^{\uparrow C}}$$

$$L_m(S_{\mathrm{mod}12}/G) = \overline{L_{am1}^{\uparrow C}} \cap \overline{L_{am2}^{\uparrow C}} \cap L_m(G) = L_{am1}^{\uparrow C} \cap L_{am2}^{\uparrow C} \supseteq (L_{am1} \cap L_{am2})^{\uparrow C} = L_{am}^{\uparrow C}$$

blocking!

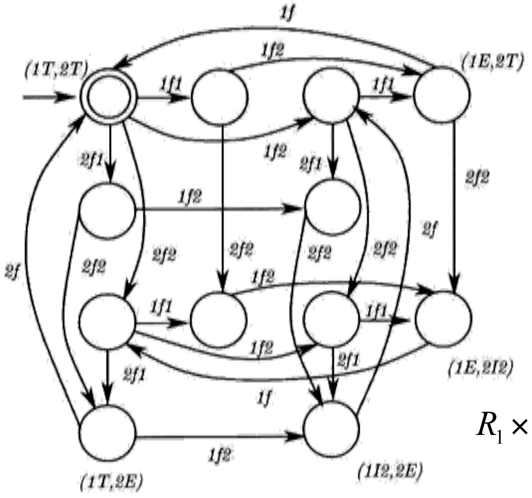
does not occur only **iff** $L_{am1}^{\uparrow C}$ and $L_{am2}^{\uparrow C}$ are nonconflicting

MODULAR CONTROL

Example: the Dining Philosophers

(reprinted from [Cassandras, Lafortune])

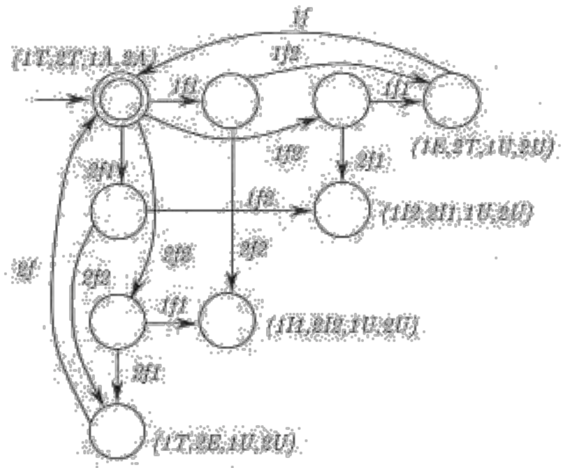
1*f*, 2*f* uncontrollable



(a)

$R_1 \times G$ represents the behavior of S_1/G

Modular Supervisor



$R_1 \times R_2 \times G$ represents the behavior of $S_{\text{mod}12}$ and it is blocking.

This is because the 2 languages $L_m(S_1|G)$ and $L_m(S_2|G)$ are conflicting. E.g.,

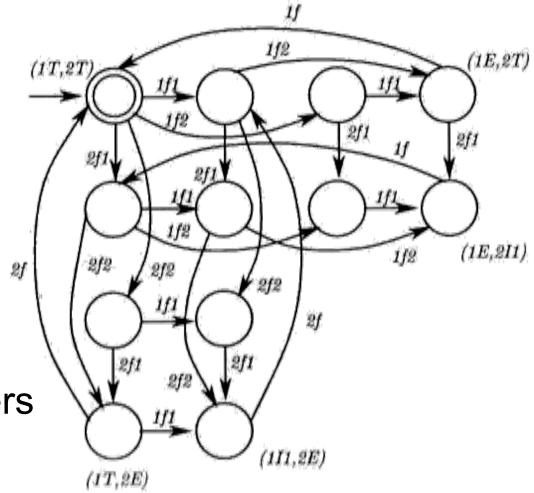
1f2 • 2f1 • 1f1 • 1f • 2f2 • 2f $\in L_m(S_1 \mid G)$

1f2 • 2f1 • 2f2 • 2f • 1f1 • 1f $\in L_m(S_2 \mid G)$

1f2 • 2f1 $\in \overline{L_m(S_1 \mid G)} \cap \overline{L_m(S_2 \mid G)}$ **but**

1f2 • 2f1 $\notin \overline{L_m(S_1 \mid G) \cap L_m(S_2 \mid G)}$

$R_2 \times G$ represents the behavior of S_2/G



(b)

Supervisors for:
(a) Fork 1 (S_1)
(b) Fork 2 (S_2)

S_i is designed to avoid fork *i* being used by both philosophers and is realized by R_i.

OBSERVABILITY CONDITION

“ If you cannot differentiate between two strings, then these strings should require the same control action ”

or

“ If you must disable an event after observing a string, then by doing so you should not disable any string that appears in the desired behavior “

DEFINITION OF OBSERVABILITY

Given $E_c, E_o \subseteq E$, and $P : E^* \rightarrow E_o^*$

$M = \overline{M}$ and K languages over event set E

K is observable with respect to $M, P,$ and E_c

if for all $s \in \overline{K}$ and for all $\sigma \in E_c$

$$(s\sigma \notin \overline{K}) \wedge (s\sigma \in M) \Rightarrow \underbrace{P^{-1}[P(s)]}_{\text{all strings with the same projection as } s}\{\sigma\} \cap \overline{K} = \emptyset$$

if this does not hold, no supervisor can differentiate between *s* and *s'* such that *P(s)=P(s')*, yet these strings may require different control actions regarding s (e.g., when $s\sigma \notin \overline{K}$ but $s'\sigma \in \overline{K}$)

When $s \in \overline{K}, s\sigma \in M, \sigma \in E_{uc}$ controllability implies that $s\sigma \in \overline{K}$, i.e., there is no need to worry about observability issues for uncontrollable events for controllable *K* w.r.t. *M* and E_{uc}

K observable **iff** $\overline{K}$ observable

CONTROLLABILITY AND OBSERVABILITY THEOREM

DES G : $G = (X, E, f, \Gamma, x_0, X_m)$

Uncontrollable events : $E_{uc} \subseteq E$

Observable events : $E_o \subseteq E$

Projection : $P : E^* \to E_o^*$

Language $K \subseteq L_m(G)$

There exists a *nonblocking* P-supervisor S_P for G such that $L_m(S_P / G) = K$ and $L(S_P / G) = \overline{K}$

iff

K is controllable with respect to $L(G)$ and E_{uc}
K is observable with respect to $L(G)$, P and E_o

K is $L_m(G)$-closed, i.e., $K = \overline{K} \cap L_m(G)$

CONTROLLABILITY AND OBSERVABILITY THEOREM

Proof is constructive:

$$S_P(t) = E_{uc} \cup \left\{ \sigma \in E_c : \exists_{s'\sigma \in \overline{K}} [P(s') = t] \right\}, \ t \in P[L(G)]$$

This supervisor enables, after string $t \in P[L(G)]$:

i. All uncontrollable events

ii. All controllable events that extend any string s', that projects to *t*, inside of $\overline{K}$

Note that i. Needs to enable only *feasible* (i.e., those enabled by $L(S_P/G)$) uncontrollable events – but this simplified the notation.

CONTROLLABILITY AND OBSERVABILITY THEOREM

Corollary

Given DES G : $G = (X, E, f, \Gamma, x_0, X_m)$

Uncontrollable events : $E_{uc} \subseteq E$

Observable events : $E_o \subseteq E$

Projection : $P : E^* \to E_o^*$

Language $K \subseteq L(G), K \neq \emptyset$

There exists a P-supervisor S_P for G such that

$$L(S_P / G) = \overline{K}$$

iff

K is *controllable* w.r. t. $L(G)$, E_{uc} and *observable* w. r. t. $L(G)$, P, E_c

OBSERVABILITY TEST

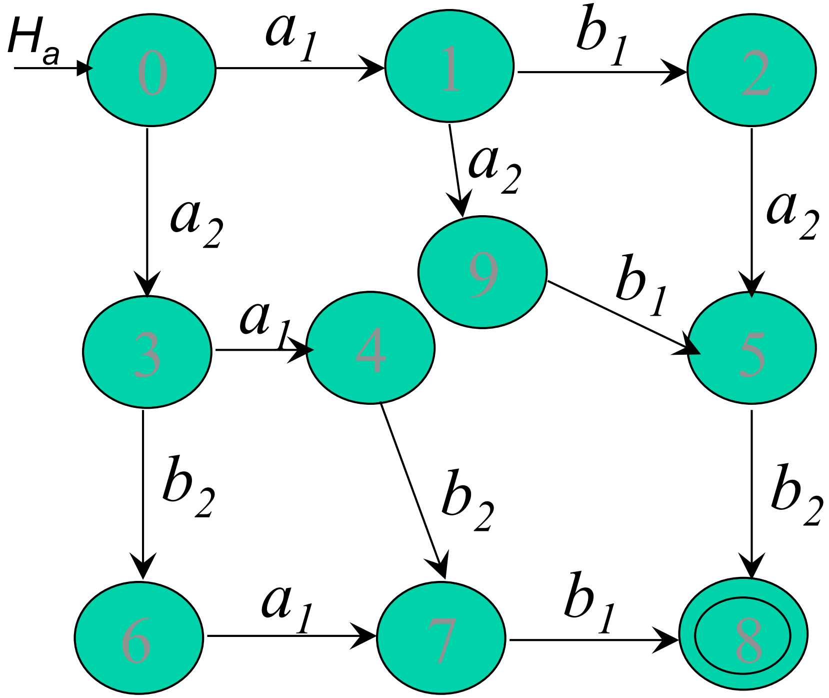
DATABASE PROBLEM

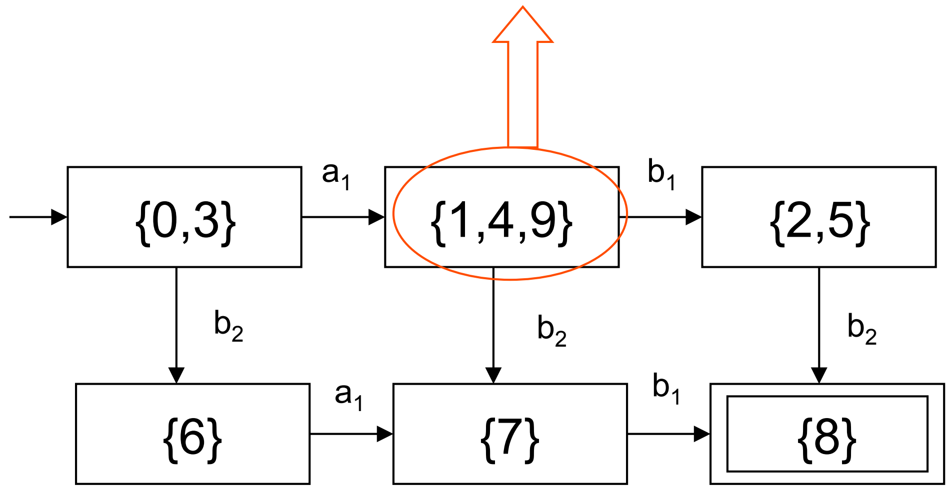
observer

$$E_{uo} = \{a_2\},\ E_c = E$$

conflict

$$\Gamma_{\{1,4,9\}} = \{b_1, b_2\}$$





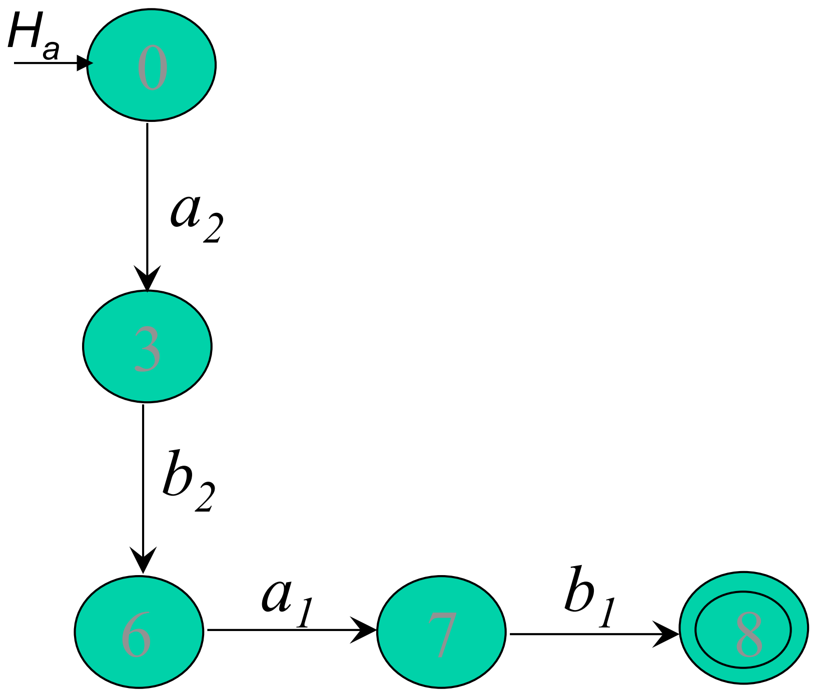
K cannot be achieved by supervisory control
even if all events are controllable

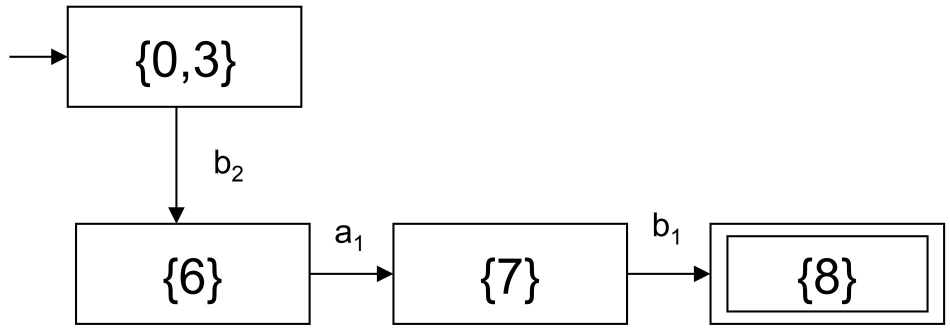
OBSERVABILITY TEST

observer

DATABASE PROBLEM
solution A

$$E_{uo} = \{a_2\},\ E_c = E$$





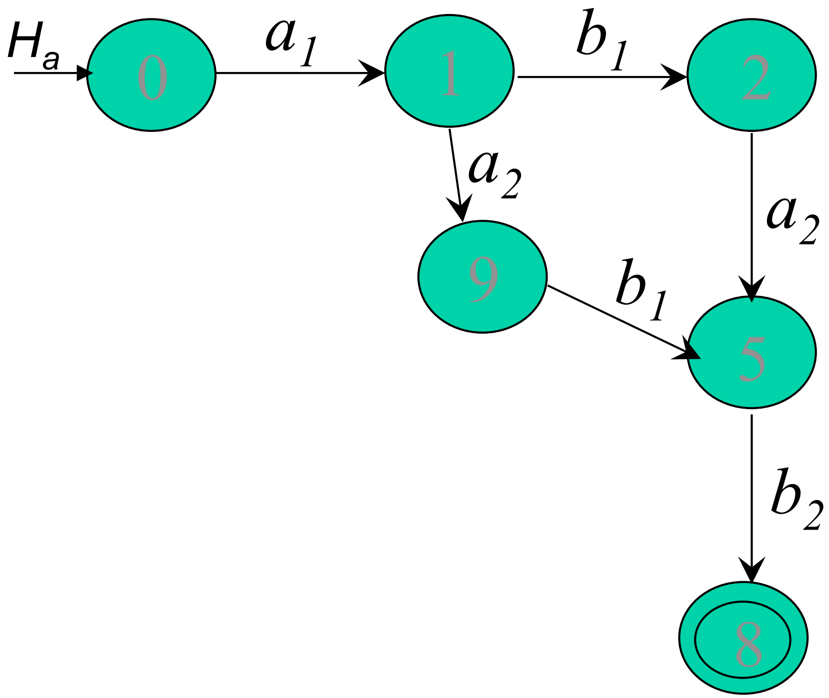
No control conflict, so
K can be achieved by supervisory control

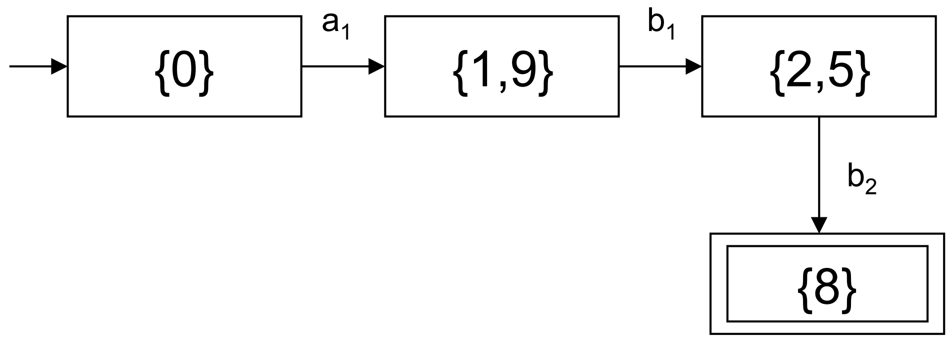
OBSERVABILITY TEST

observer

DATABASE PROBLEM
solution B

$$E_{uo} = \{a_2\},\ E_c = E$$





No control conflict, so
K can be achieved by supervisory control

REALIZATION OF P-SUPERVISORS

If $K \subseteq L(G)$ is controllable and observable, the COT tells us that the P-supervisor S_P defined by

$$S_P(t) = E_{uc} \cup \{\sigma \in E_c : \exists_{s'\sigma \in \overline{K}} [P(s') = t]\}, \; t \in P[L(G)]$$

results in $L(S_P / G) = \overline{K}$, excluding $\overline{K} = L(G)$ and $\overline{K} = \emptyset$.

Again, we restrict S_P to be *realized* by an FSA

We will be dealing with regular languages *L(G)* and *K*, with finite, thus implementable, realizations.

REALIZATION OF P-SUPERVISORS

1. Build a trim automaton R that generates and marks the language $\overline{K}$. The event set of R is E and E_o is the subset of observable events;
2. Build R_{obs}, the observer for R corresponding to the set E_o;

The active event set of R_{obs} does not necessarily encode the set of events enabled by S_P, since it does not contain any information on what to do with events in E_{uo}.

3. Let t be the current string of *observable* events and let $x_{obs,current}$ be the state of R_{obs} after t (i.e., after the last observable event in t but before the next observable event, R could be in any of the states in the set $x_{obs,current}$);
4. Then $S_P^{realized}(t) = \bigcup_{x \in x_{obs,current}} \Gamma_R(x)$, where Γ_R is the active event function of R.

Note that $S_P^{realized}$ enables only *feasible* uncontrollable events, while in COT, S_P enables *all* uncontrollable events, for the sake of a simpler notation.

$S_P^{realized}$ is admissible (since $\overline{K}$ is controllablle), and $S_P^{realized}(t) \cap E_c = S_P(t) \cap E_c$.
It is not necessary to store R, since we can pre-compute all the enabled events for each state of R_{obs}.

Properties of observability

If K_1 and K_2 are observable, then $K_1 \cup K_2$ need not be observable.

If K_1 and K_2 are prefix-closed and observable, then $K_1 \cap K_2$ is prefix-closed and observable.

$E = E_c = \{\alpha, \beta\}$ and $E_o = \{\beta\}$

$M = \{\varepsilon, \alpha, \beta, \alpha\beta\},\ K_1 = \{\alpha\},\ K_2 = \{\beta\}$

K_1 and K_2 are observable, but $\mathrm{K} = K_1 \cup K_2 = \{\alpha, \beta\}$ is not.

E.g., $s = \alpha, s' = \varepsilon, \sigma = \beta \in E_c$

then $s\sigma \notin \overline{K}, s\sigma \in M, s'\sigma \in \overline{K}, s'\sigma \in M$

but $s'\sigma \in P^{-1}[P(s)]\sigma$ since $P(s) = P(s')$.

INFIMAL PREFIX-CLOSED OBSERVABLE SUPERLANGUAGE

$$CO_{out}(K) = \left\{ L \subseteq E^* : (K \subseteq L \subseteq M) \text{ and } (\overline{L} = L) \text{ and } L \text{ is observable} \right\}$$

Existence

We would like to find the "smallest" superlanguage of K which is observable with respect to fixed M, E_o and E_c.

Q: Does it exist?

A: Yes!

$$K^{\downarrow O} = \bigcap_{L \in CO_{out}(K)} L$$

By definition, $K^{\downarrow O} \subseteq L$ and is not empty because $M \in CO_{out}(K)$, for any $L \in CO_{out}(K)$ $\Rightarrow$

$\Rightarrow$ $K^{\downarrow O}$ is the *infimal prefix-closed observable superlanguage* of K and belongs to $CO_{out}(K)$

- In the "worst" case, $K^{\downarrow O} = M$
- If K is observable, then $K^{\downarrow O} = \overline{K}$

OBSERVABILITY, CONTROLLABILITY AND INTERSECTION

The results about $\downarrow C$ and $\downarrow O$ can be combined to conclude that the *infimal prefix-closed observable and controllable superlanguage* of a given language does exist.

$$CCO_{out}(K) = CC_{out}(K) \cap CO_{out}(K)$$

$CCO_{out}(K)$ contains the superlanguages of K that are prefix-closed, controllable and observable. $CCO_{out}(K)$ is closed under arbitrary intersections, therefore its infimal element exists and is the *infimal prefix-closed observable and controllable superlanguage* of K, denoted as $K^{\downarrow CO}$.

- In the “worst” case, $K^{\downarrow CO} = M$
- If K and M are regular, $K^{\downarrow O}$ and $K^{\downarrow CO}$ are regular (there are formulas to compute them)

SUPERVISORY CONTROL PROBLEMS UNDER PARTIAL OBSERVATION

for a DES G with event set E and $E_o \subseteq E, P : E^* \to E_o^*, E_{uc} \subseteq E$ and $L_a = \overline{L}_a \subseteq L(G)$

Basic Supervisory Control and Observation Problem (BSCOP)

Find a supervisor S_P such that:

1. $L(S_P | G) \subseteq L_a$
2. $L(S_P | G)$ is "the largest it can be", i.e., for any other supervisor S_{other} such that $L(S_{other} | G) \subseteq L_a$, $L(S_{other} | G) \subseteq L(S_P | G)$

for a DES G with event set E and $E_o \subseteq E, P : E^* \to E_o^*, E_{uc} \subseteq E$ and $L_a = \overline{L}_a \subseteq L(G)$ and admissible marked language $L_{am} \subseteq L_m(G)$, L_{am} is $L_m(G)$-closed

BSCOP - Nonblocking (BSCOP-NB)

Find a *nonblocking* P-supervisor S such that:

1. $L_m(S_P | G) \subseteq L_{am}$
2. $L_m(S_P | G)$ is "the largest it can be", i.e., for any other P-supervisor S_{other} such that $L_m(S_{other} | G) \subseteq L_{am}$, $L_m(S_{other} | G) \subseteq L_m(S_p | G)$

SUPERVISORY CONTROL PROBLEMS UNDER PARTIAL OBSERVATION

Due to the results on observability and union, the supremal observable sublanguage of a given language need not exist. Therefore, the supremal observable and controllable sublanguage of a given language need not exist.

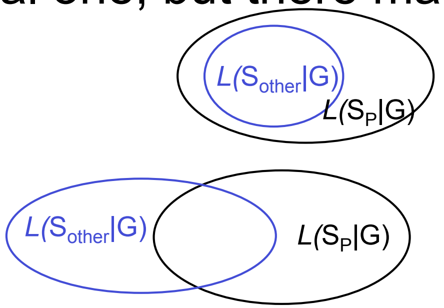
⇒In general, there is no solution for BSCOP and BSCOP-NB that satisfies requirement 2. of both problems.

One possible approach to overcome this difficulty is to calculate *maximal* (w. r. t. set inclusion) observable and controllable sublanguages of L_a (and L_{am}).

By maximal we mean that there is no other observable and controllable sublanguage *strictly larger* than the maximal one, but there may be other *incomparable* maximals.

In that case, 2. is replaced by the weaker

2'. $L(S_{other} | G) \subseteq L_a \Rightarrow L(S_P | G) \not\subset L(S_{other} | G)$



SUPERVISORY CONTROL PROBLEMS UNDER PARTIAL OBSERVATION

for a DES *G* with event set *E* $E_o \subseteq E, P: E^* \rightarrow E_o^*, E_{uc} \subseteq E$ and $L_r = \overline{L_r} \subseteq L(G)$ and

Dual of BSCOP (DuSCOP)

Find a P-supervisor S_P such that:

1. $L(S_P \mid G) \supseteq L_r$
2. $L(S_P \mid G)$ is "the smallest it can be", i.e., for any other supervisor S_{other} such that $L(S_{other} \mid G) \supseteq L_r$, $L(S_{other} \mid G) \supseteq L(S_P \mid G)$

Solution: $L(S_P \mid G) = L_r^{\downarrow CO}$

Note that L_r need not be prefix-closed and could be given as a subset of $L_m(G)$

THE PROPERTY OF NORMALITY

Consider $M = \overline{M} \subseteq E^*$, and $P : E^* \rightarrow E_o^*$

$K \subseteq M$ is said to be *normal* w. r. t. *M* and *P* if

$$\overline{K} = P^{-1}[P(\overline{K})] \cap M.$$

i.e., $\overline{K}$ can be exactly recovered from its projection $P(\overline{K})$ and from M.

$\emptyset$ and M are both normal.

$\overline{K} \subseteq P^{-1}[P(\overline{K})] \cap M$ always holds.

THE PROPERTY OF NORMALITY

Normality and Observability

If $K \subseteq M$ is normal w. r. t. M and P,
then K is observable w. r. t. to M, P and $E_c, \forall E_c \subseteq E$.
However, the converse statement is not true in general.

Normality $\Rightarrow$ Observability

Normality and Union

If $K_1, K_2 \subseteq M$ are normal w. r. t. M, then so is $K_1 \cup K_2$.

Normality is preserved under union

therefore, we can establish the existence of:

- the *supremal normal sublanguage* of K, denoted as $K^{\uparrow N}$
- the *supremal controllable and normal sublanguage* of K, denoted as $K^{\uparrow CN}$

THE PROPERTY OF NORMALITY

Equivalence of Normality and Observability

Assume that $E_c \subseteq E_o$. If K is controllable w. r. t. M and E_{uc}, and observable w. r. t. to M, P and E_c, then K is normal w. r. t. M, P.

when all the controllable events are observable, or equivalently, when all the unobservable events are uncontrollable, the intrinsic difficulties associated with observability and in particular the lack of existence of a supremal observable sublanguage are alleviated if controllability enters the picture. This is because controllability will "take care of" some of the unobservable events and "reduce" observability to normality, a better behaved property.

In these cases, BSCOP and BSCOP-NB do have "optimal" solutions

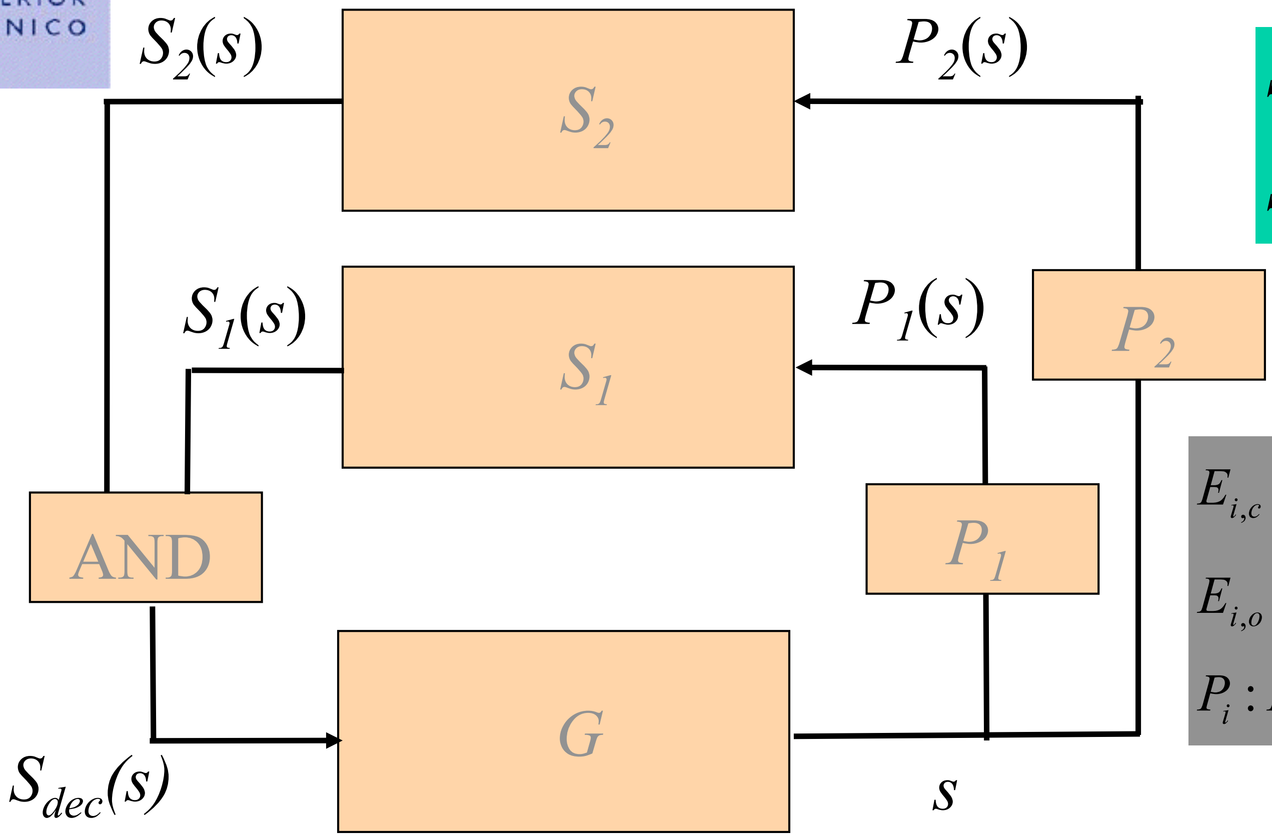
THE PROPERTY OF NORMALITY

Properties of Normality

- When K and M are regular, then so are $K^{\uparrow N}$ and $K^{\uparrow CN}$
- If K is prefix-closed, then so are $K^{\uparrow N}$ and $K^{\uparrow CN}$
- If K is $L_m(G)$-closed, then so are $K^{\uparrow N}$ and $K^{\uparrow CN}$ (useful for BSCOP-NB)

$K^{\uparrow CN}$ provides a sub-optimal solution to BSCOP and BSCOP-NB – it meets requirement 1 but not necessarily 2. This solution may not be maximal in general, i.e., there may be CO languages that are strictly larger than the supremal CN sublanguage.

DECENTRALIZED CONTROL



$$S_i(s) = S_{P_i}[P_i(s)]$$

$$S_{dec}(s) = S_1(s) \cap S_2(s)$$

$$E_{i,c} \subseteq E_c, \quad \bigcup_{i=1}^{n} E_{i,c} = E_c$$

$$E_{i,o} \subseteq E_o, \quad \bigcup_{i=1}^{n} E_{i,o} = E_o$$

$$P_i : E^* \rightarrow E_{i,o}^* \text{ corresponding to } E_{i,o}$$

Global behavior is described by $L(S_{dec}/G)$.

"Local" behaviors are described by $P_i[L(S_{dec}/G)]$

DECENTRALIZED CONTROL

Q.: Given a "target" language K that restricts the global behavior $L(G)$, what is the *necessary and sufficient condition* on K, beyond controllability, that will ensure the existence of S_i, i=1,...,n, such that $L(S_{dec}/G) = K$?

H.: the condition must be *stronger than*

K is observable w. r. t. $L(G)$, E_o and E_c

since *if the centralized problem cannot be solved, neither can the decentralized problem.*

However, it should be *weaker than*

K is observable w. r. t. $L(G)$, P_i and $E_{i,c}$, i=1,...,n

since there may be events that can be controlled by more than one supervisor, therefore we may not need full "local" observability at all sites. The supervisors may be able to "share the work" on the *common* controllable events, in the sense that no single supervisor is uniquely responsible for disabling these events. Which supervisor disables a common event could depend on the string of events executed so far by G.

CO-OBSERVABILITY

Let K and $M = \overline{M}$ be languages over event set E.

$$E_{i,o}, E_{i,c} \subseteq E, P_i : E^* \to E_{i,o}^*, i = 1, \ldots, n$$

K is said to be *co-observable* w. r. t. *M*, P_i and $E_{i,c}$ if, for all $s \in \overline{K}$ and for all $\sigma \in E_c$

$$\left(s\sigma \notin \overline{K}\right) \wedge \left(s\sigma \in M\right) \Rightarrow$$

$$\exists_{i \in \{1,\ldots,n\}} : P_i^{-1}\left[P_i(s)\right]\{\sigma\} \cap \overline{K} = \varnothing \wedge \sigma \in E_{i,c}.$$

If event σ needs to be disabled, then at least one of the supervisors that can control σ must unambiguously know that it must disable σ, that is, from this supervisor's viewpoint, disabling σ does not prevent any string in $\overline{K}$; consequently, each supervisor can still follow the "pass the buck" policy.

CO-OBSERVABILITY

If $E_{i,o} = E_o, E_{i,c} = E_c$ and

$E_{j,o} = E_{j,c} = \varnothing, j = 1, \ldots, n; j \neq i$

then co-observability reduces to observability

If $E_{i,c} \cap E_{j,c} = \varnothing \quad i, j = 1, \ldots, n;$

then passing the buck does not apply and

co-observability of K is equivalent to

K is observable w.r.t. L(G), P_i and E_{ic} for each i=1,...,n

CONTROLLABILITY AND CO-OBSERVABILITY THEOREM

DES G : $G = (X, E, f, \Gamma, x_0, X_m)$

Uncontrollable events : $E_{i,c}$, $E_c = E \setminus E_{uc} \subseteq E$

Observable events : $E_{i,o}$, $E_o \subseteq E$

Projection : $P_i : E^* \to E_{i,o}^*$, $i = 1, \ldots, n$

Language $K \subseteq L_m(G)$, $K \neq \emptyset$

There exists a *nonblocking* decentralized supervisor S_{dec} for G such that $L_m(S_{dec}/G) = K$ and $L(S_{dec}/G) = \overline{K}$

iff

K is controllable with respect to $L(G)$ and E_{uc}

K is co-observable with respect to $L(G)$, P_i and $E_{i,c}$, i=1,...,n

K is $L_m(G)$-closed

Proof is constructive:

$$S_i(s) = S_{P_i}(s_i) = E_{i,uc} \cup \{\sigma \in E_{i,c} : \exists_{s'\sigma \in \overline{K}} [P_i(s') = s_i]\} \; s \in L(G), P_i(s) = s_i$$

SUPERVISORY CONTROL

Further reading

- Reduced-state realization of supervisors
- Algorithms to compute $K^{\uparrow C}$ and $K^{\downarrow C}$
- SCPB Supervisory Control problem

Other references

- *Supervisory Control of Discrete Event Systems Using Petri Nets*, J. O. Moody, P. J. Antsaklis, Kluwer Academic Publ., 1998 (ISR)
- "The Control of Discrete Event Systems", P. J. Ramadge, W. M. Wonham, *Proceedings of the IEEE*, Vol. 77, No. 1, pp. 81-98, January 1989

Acknowledgments to Dejan Milutinovic, who helped preparing some of the slides in this chapter, for a few sessions of an ISR/IST Reading Group on DES and of ISR/IST Control Theory Group.